3-1-2008

Implementation and Validation of the NEES Hybrid Simulation Infrastructure at Lehigh University's RTMD Facility

Thomas Marullo

Cheng Chen

Jun Cao

James M. Ricles

Implementation and Validation of the NEES Hybrid Simulation Infrastructure at Lehigh University's RTMD Facility

by

Thomas Marullo

Cheng Chen

Jun Cao

James M. Ricles

ATLSS Report No. 08-02

March 2008

ATLSS is a National Center for Engineering Research on Advanced Technology for Large Structural Systems

117 ATLSS Drive
Bethlehem, PA 18015-4729

Phone: (610)758-3525
Fax: (610)758-5902

www.atlss.lehigh.edu
Email: inatl@lehigh.edu

Implementation and Validation of the NEES Hybrid Simulation Infrastructure at Lehigh University's RTMD Facility

by

Thomas Marullo
Research Scientist

Cheng Chen
Postdoctoral Research Associate

Jun Cao
Research Scientist

James M. Ricles
Bruce G. Johnston Professor

ATLSS Report No. 08-02

March 2008

ATLSS is a National Center for Engineering Research on Advanced Technology for Large Structural Systems

117 ATLSS Drive
Bethlehem, PA 18015-4729

Phone: (610)758-3525
Fax: (610)758-5902

www.atlss.lehigh.edu
Email: inatl@lehigh.edu

Acknowledgements

The following persons and sites participated in this study: Andreas Schellenberg and Catherine Whyte at UC Berkeley who provided OpenFresco support; Gary Haussmann at the University of Colorado at Boulder who provided distributed testing support; Kyu-Sik Park at the University of Illinois at Urbana-Champaign who provided distributed testing support; Professor Richard Christenson at the University of Connecticut for distributed testing support; and Min-woo Chang at Seoul National University for distributed testing support. The participation and assistance of these individuals is greatly appreciated. The study documented in this study was funded by NEESinc under the cooperative agreement established by the National Science Foundation (NSF) under Grant No. CMS-0402490 within the George E. Brown, Jr. Network for Earthquake Engineering Simulation Consortium Operation.

Table of Contents

Abstract

Software packages for performing local, distributed and real-time hybrid simulations exist within the NEES community and are widely used. As a leader in real-time hybrid simulations, Lehigh University's Real-Time Multi-Directional (RTMD) Earthquake Simulation Facility (RTMD) NEES equipment site has installed and validated these existing tools to extend the hybrid capabilities of this facility. These hybrid tools, called OpenFresco and UI-SimCor, create a layer of abstraction which makes hybrid testing more efficient and easier to perform. Several validation tests, both local and distributed, were performed at the RTMD facility in conjunction with analysis of the real-time capabilities of the existing software and recommendations to the community.

1 Introduction and Background

This document describes tasks required to install and validate OpenFresco with OpenSees and UI-SimCor at the RTMD facility at Lehigh University for use in hybrid simulation. The objectives were: (1) to implement and validate OpenFresco at the RTMD facility at Lehigh University to perform conventional (i.e., slow) local hybrid and distributed hybrid simulation; (2) to implement and validate UI-SimCor at the RTMD facility to perform conventional local hybrid and distributed hybrid simulation; and (3) to identify gaps and provide recommendations for future developments in OpenFresco and UI-SimCor that enable this software to perform real-time hybrid simulation at the RTMD facility. To accomplish these objectives, a series of seven tasks were performed. The completion of these tasks ensures RTMD's compatibility with other NEES and non-NEES sites that utilize either OpenFresco or UI-SimCor to conduct distributed hybrid simulation.

Hybrid testing combines physical testing with numerical simulation (Dermitzakis and Mahin 1985), and provides a viable alternative for dynamic testing of structural systems. The structure to be tested is divided into a physical component (test structure) and a numerical model (analytical substructure(s)). The analytical substructure(s) includes the mass of the structure (lumped at discrete locations), and the inherent structural damping. During the test, the displacement response of the structure is calculated using time step integration of the equations of motion. The displacements are imposed on the test structure using actuators and to the analytical substructure at the discrete locations where lumped masses are assumed. The forces required to produce these displacements in the test structure and analytical substructure(s) are measured and computed, respectively, and fed back to the simulation to calculate the command displacements corresponding to the next time step.

The RTMD facility has the need to implement OpenFresco, version 2.0 with OpenSees version 1.7.4 and UI-SimCor version 2.6 to become fully compatible and able to participate in distribute hybrid testing with other equipment sites that utilize this software. These software packages will have to be integrated into the system architecture for the RTMD facility that is described below.

2 RTMD Facility

The Lehigh RTMD NEES Equipment Site is a large-scale experimental laboratory facility with real-time local hybrid simulation capabilities. The facility is housed in the Multi-directional Experimental Laboratory at the ATLSS Engineering Research Center at Lehigh University. The ATLSS Laboratory has a strong floor that measures 31.1m x 15.2 m in plane, and a multi-directional reaction wall that measures up to 15.2 m in height. Anchor points are spaced on a 1.5-m grid along the floor and walls. Each anchor point can resist 1.33 MN tension force and 2.22 MN shear force. Additional steel framing is used in combination with the strong floor and reaction walls to create a wide variety of test configurations. To create the RTMD earthquake simulation facility, several pieces of equipment have been installed in the ATLSS Laboratory. This equipment includes five dynamic, double-rodded hydraulic actuators with a +/-500 mm stroke. Two of these

actuators have a 2300 kN maximum load capacity, and the remaining three actuators have 1700 kN maximum load capacity. Each of the actuators is ported for three 1500 liter/min servo-valves, enabling them to achieve a maximum nominal velocity of 840 mm/sec (2300 kN actuators) and 1140 mm/sec (1700 kN actuators). The existing hydraulic power supply system at ATLSS consisted of five 2250 liter/min pumps. A hydraulic oil reserve and two banks of accumulators were added to enable strong ground motion effects to be sustained for up to 30 seconds. The accumulators supply a total accumulated oil volume of 3030 liters.

The real-time integrated control architecture is given below in Figure 2-1. An 8-channel digital controller (identified as the Real-time Control Workstation in Figure 2-1, with a 1024 Hz clock speed, controls the motion of the actuators through a closed servo-control loop. The Real-time Control Workstation is integrated with the dual Pentium 4 Xeon 2.4 GHz Simulation Workstation, Real-Time Target Workstation and Data Acquisition Mainframe, as well as the Telepresence Server using SCRAMNet*. SCRAMNet is a fiber optic communication device that enables shared memory and time synchronization to the Control, Simulation and Target Workstations. The Target Workstation communicates with the Control Workstation and Data Acquisition Mainframe using SCRAMNet, thereby providing a single synchronization source for experiments. The Data Acquisition Workstation controls a high speed 256-channel data acquisition mainframe capable of acquiring data at 1024 Hz per channel. The integrated control system configuration permits complex testing algorithms, servo-hydraulic control laws, and analytical substructures to be developed on the Simulation Workstation and downloaded onto the Target Workstation. The latter is used for real-time hybrid testing, where Mathworks Simulink and Real-Time Workshop are used to create the analytical substructures. The Target Workstation runs Mathworks xPC Target software. The testing algorithms and any new servo-control laws are developed using Simulink, compiled on the Simulation Workstation and downloaded to the Target Workstation. Command signals for imposing displacements on a test structure are generated on the Target Workstation by the integration algorithm, where complex analytical models can reside (e.g., MATLAB or Simulink) for integrating the equations of motion in conjunction with the test structure for real-time hybrid testing. Feedback signals needed to determine the command signal for the next time step during a test are acquired from the Control Workstation and the Data Acquisition Mainframe (e.g., the measured actuator forces and current position of the test structure to enable kinematic compensation for multi-directional real-time pseudo-dynamic tests).

Slow hybrid testing can be performed using two different procedures at the RTMD facility. By the use of a ramp generator with an expanded time scale placed on the Real-time Target Workstation, hybrid simulations can be slowed down while still remaining deterministic. The second method involves a non-deterministic method of using control algorithms developed in higher level software environments such as MATLAB, LabVIEW and Java applications on the Simulation Workstation. The Simulation Workstation then communicates with the Real-time Control Workstation via SCRAMNet. Distributed hybrid simulation is currently conducted through the use of communication software (NTCP) on the Simulation Workstation in conjunction with MATLAB to create

* SCRAMNet is a real-time communications network, based on a replicated, shared-memory (reflective memory) concept. http://www.cwcembedded.com/products/0/1/71.html

an analytical substructure., Real-time Simulink models of the servo-hydraulic control system and actuators can be used in either slow or fast-hybrid simulation to emulate a hybrid simulation in hydraulics-off mode. Hydraulics-off mode is used at the RTMD facility in pre-test simulations to verify algorithms, control gains, demand on equipment, and for training.

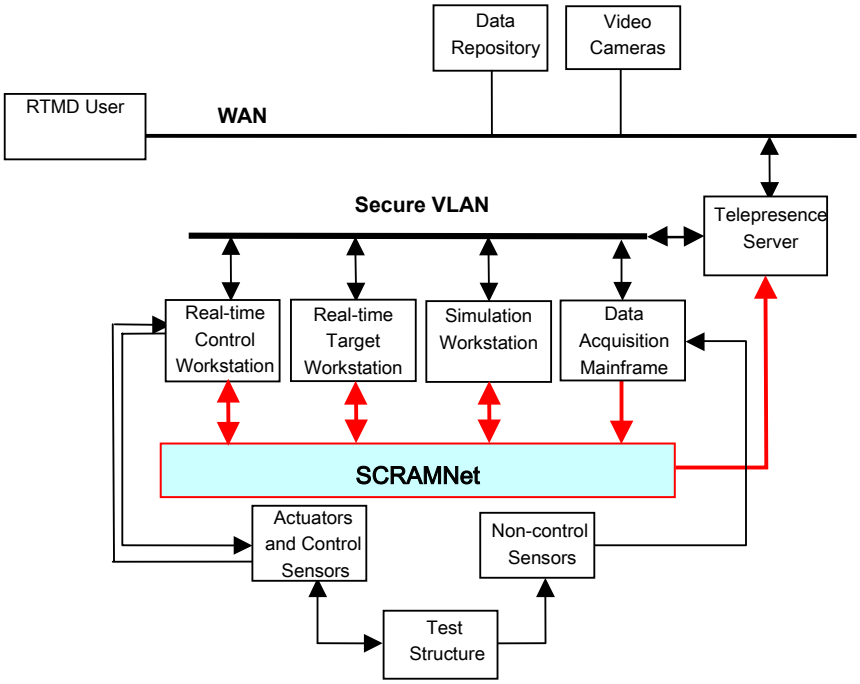


Figure 2-1 RTMD Seismic Simulation Facility integrated control system architecture

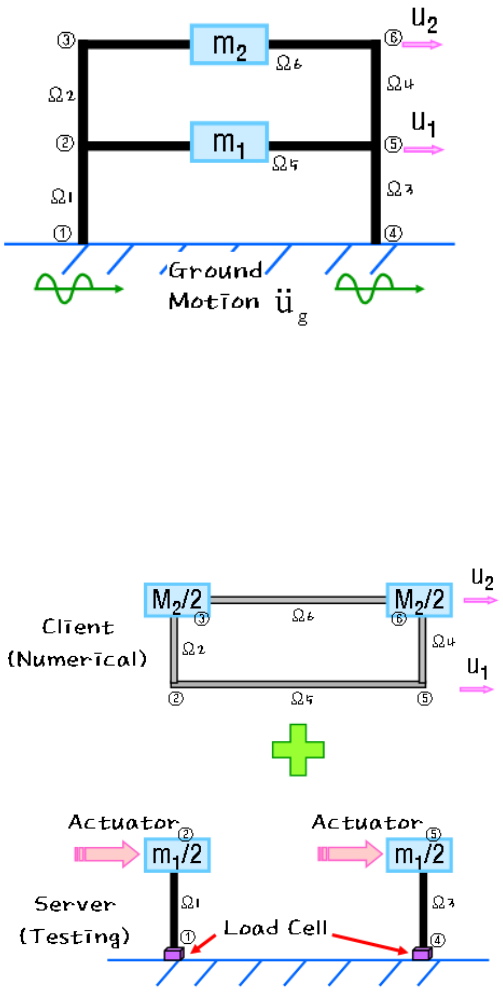
3 Activities

Listed below are the activities that were performed which ensures the RTMD has fully validated and evaluated the NEES hybrid simulation infrastructure containing OpenFresco version 2.0, OpenSees version 1.7.4 and UI-SimCor version 2.6. The activities included conducting seven phases of validation tests.

- *Validation Tests Phase 1:* Evaluate OpenFresco and OpenSees without hydraulic system power
- *Validation Tests Phase 2:* Evaluate OpenFresco and UI-SimCor without hydraulic system power
- *Validation Tests Phase 3:* Evaluate OpenFresco and OpenSees with single actuator control
- *Validation Tests Phase 4:* Evaluate OpenFresco and UI-SimCor with single actuator control
- *Validation Tests Phase 5:* Evaluate OpenFresco and OpenSees for three-site distributed simulation with multiple actuator control
- *Validation Tests Phase 6:* Incorporate new explicit integration algorithm into UI-SimCor using OpenFresco for local hybrid simulation

- *Validation Tests Phase 7:* Evaluate real-time capabilities of OpenFresco for use at the RTMD facility

For the OpenSees-based simulations, a two degree of freedom (DOF) steel frame structure was modeled using six beam-column type elements (see Figure 3-1). The two degrees of freedom are identified in Figure 3-1 as U1 and U2, and are the lateral displacements at the first and second floors of the structure, respectively, Elements 1 and 3 in the model (which are the first story column in the frame) were independently modeled as separate OpenFresco beam-column element servers with a transverse stiffness of 505 kips/in each. The structure was subjected to ground motions caused by the 1940 El Centro earthquake.



2 DOF Steel Frame Structure	
Column	W14×311
Beam	W36×150
Height of Story	144 (in.)
Length of Beam	360 (in)
Ground Motion	El-Centro, 1940, NS Component
Damping	Rayleigh Proportional
Damping ratio	2% (at 1, 2nd mode)
Mass - m_1 / m_2	11.964/8.232 (kip·s^2/in.)
Period - T_1 / T_2	0.976 / 0.397 (sec)
Time Step, Δt	0.01 (sec)
Total Steps	4000

Figure 3-1 Steel frame structure used for OpenSees model

The UI-SimCor model could not be based on the same model used in OpenSees because UI-SimCor is limited to one DOF when using OpenFresco. This limitation resulted in using a bridge model with two piers, where the piers were not coupled (Figure 3-2). The mass properties were the same as in the OpenSees model. Each pier was modeled as a beam-column element in OpenFresco with a transverse stiffness of 1010 kips/in.

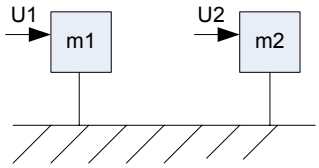


Figure 3-2 UI-SimCor bridge pier model

The final phase of the validation tests was intended to identify gaps that exist which prevent OpenFresco and UI-SimCor from performing deterministic real-time local or distributed hybrid simulation at the RTMD facility. Recommendations are provided in this report for future developments to close these gaps.

The validation tests discussed above were motivated by scenarios associated with typical research projects that would be conducted at the RTMD facility. The most common use-case scenarios include: (1) performing a distributed hybrid simulation that is coordinated by a remote site, where this site performs analysis and issues commands to an experimental substructure(s) located at the RTMD facility; (2) performing a distributed hybrid simulation that is coordinated by the RTMD facility, where the RTMD facility performs analysis and issues commands to experimental substructures located at the RTMD facility and at one or more remote facilities.

4 Validation Tests

4.1 Validation Test Phase 1

Validation Test Phase 1 involved performing multiple two server hybrid simulations where all sites were running in a hydraulics-off mode. The setup included a client program using OpenSees and OpenFresco and two server programs using OpenFresco. The client program used the Newmark Explicit integration algorithm and numerically modeled the upper story of the two-story steel structure shown in Figure 3-1. As noted in Figure 3-1, the time step was equal to 0.01 secs. For each time step the command displacements were sent to servers 1 and 2, and the corresponding restoring forces developed in the models of the first story columns on these servers were sent back to the client. Each server modeled a single column in the first story. To provide baseline test duration and iteration times that all the tests can be compared to, Lehigh University performed a local test that included two PCs on the Lehigh network. One PC acted as the client and one server while the other PC acted as the second server.

Validation Test Phase 1 involved performing the five tests summarized in Table 4-1. The locations of the sites are identified for each test in this table. For Test 1A, Lehigh University acted as the client running OpenSees with both servers running OpenFresco, as portrayed in the schematic given in Figure 4-1. For Test 1B, Lehigh University acted as the client running OpenSees with both servers running OpenFresco. As shown in Figure 4-2, in Test 1B the RTMD Simulation Workstation acted as server 2 and was running OpenFresco with the SCRAMNet experimental control activated which sent commands to the RTMD Real-time Target Workstation and received a feedback restoring force. The Real-time Target Workstation was running a hydraulics-off model of a single

actuator producing a linear-elastic feedback restoring force. A MATLAB middleware program was developed to communicate between the SCRAMNet experimental control module in OpenFresco and the Real-time Target Workstation due to a mismatching in the SCRAMNet memory mapping and data types.

For Test 1C, Lehigh University acted as the client running OpenSees with server 1 running OpenFresco (see Figure 4-3). The University of California, Berkeley (UC Berkeley) acted as the second server. For Test 1D, Lehigh University acted as the client running OpenSees with server 1 running OpenFresco, while Seoul National University (SNU) acted as the second server (see Figure 4-3). For Test 1E, SNU acted as the client running OpenSees and Lehigh University acted as both servers running OpenFresco (see Figure 4-3).

Test	Client	Server 1	Server 2
1A	Lehigh	Lehigh	Lehigh
1B	Lehigh	Lehigh	Lehigh SCRAMNet
1C	Lehigh	Lehigh	UC Berkeley
1D	Lehigh	Lehigh	SNU
1E	SNU	Lehigh	Lehigh

Table 4-1 Validation Test Phase 1 test matrix

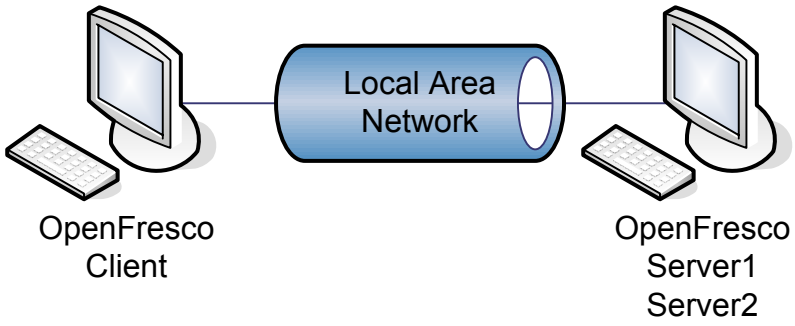


Figure 4-1 Local hybrid simulation setup, Test 1A

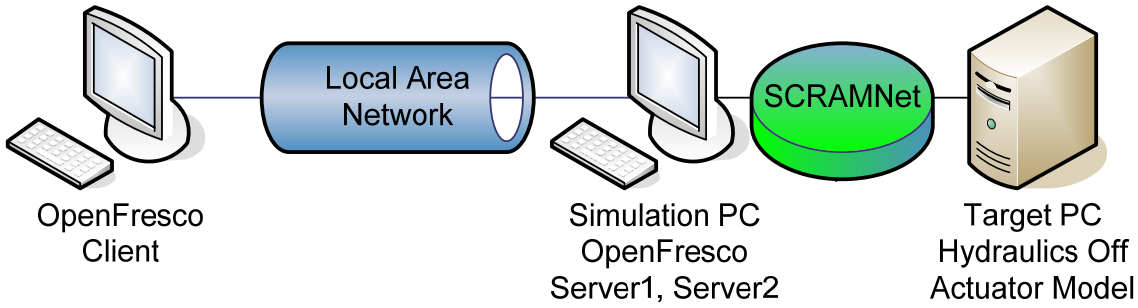


Figure 4-2 Local hybrid simulation setup using SCRAMNet and hydraulics off mode, Test 1B

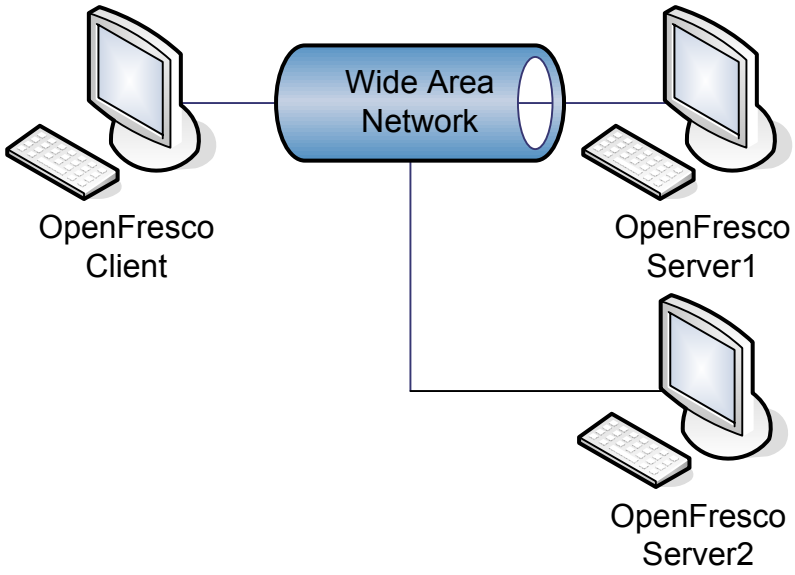


Figure 4-3 Distributed hybrid simulation setup, Tests 1C, 1D, and 1E

The duration of each time step in all tests was recorded along with the total time for the test to be completed. Table 4-2 compares the physical distance between Lehigh University and the remote sites, UC Berkeley and SNU, along with the average time per time step and total time for the tests. In Tests 1C and 1D, as noted above Lehigh University was both the client and one server; therefore only one long distance round trip transfer had to be made per time step. For Test 1E, because SNU was the client and Lehigh University acted as both servers, two long distance round trip transfers had to be made per time step, each one being associated with a server. This led to the approximate doubling of the duration for an average step time and total time of the simulation.

Test	Distance (km)	Average time per time step (ms)	Total time for simulation (min)
1A	0.03	23	1.51
1B	0.03	29	1.92
1C	4000	363	24.21
1D	11000	902	60.15
1E	11000	1762	117.45

Table 4-2 Distance and duration comparison for Validation Test Phase 1

For all five tests, the results from the hybrid simulations were identical for the displacement responses for U1 and U2, respectively, when compared with a time history analysis of the same structure using OpenSees (see Figures 4-4 and 4-5).

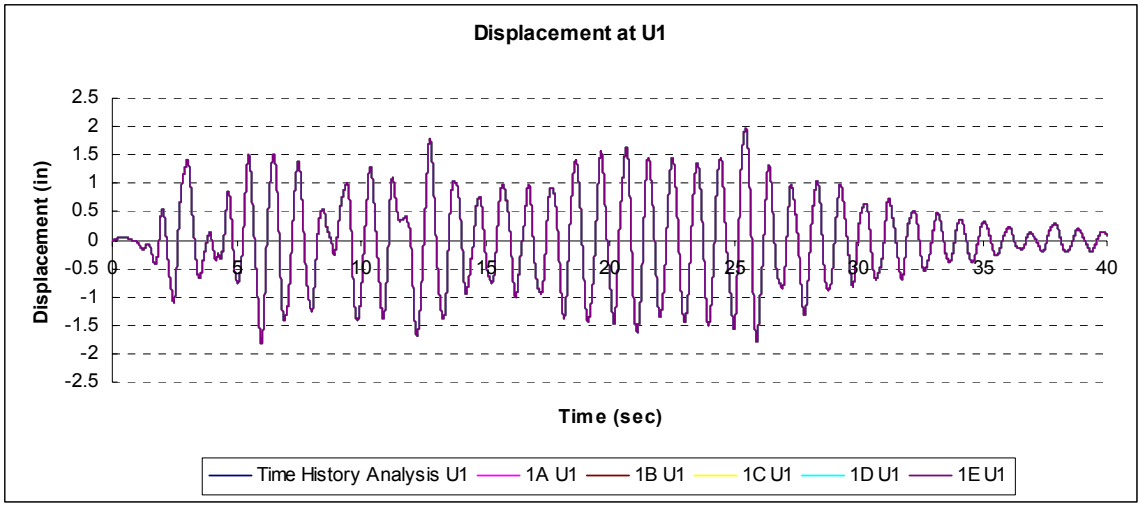


Figure 4-4 Validation Test Phase 1 – comparison of displacement response at DOF U1

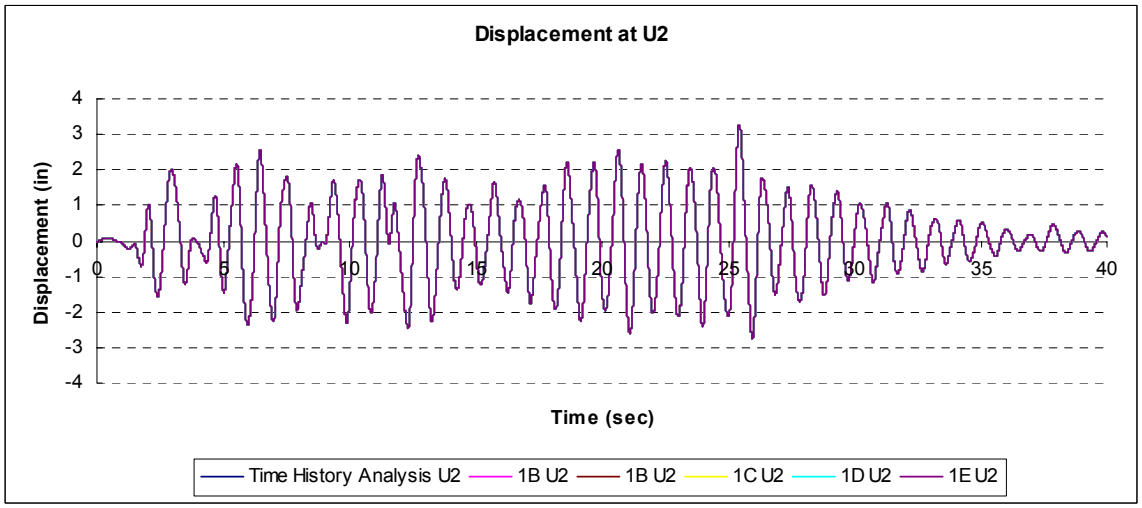


Figure 4-5 Validation Test Phase 1 – comparison of displacement response at DOF U2

4.2 Validation Test Phase 2

Validation Test Phase 2 involved performing multiple two server hybrid simulations, where all sites that were involved were running in a hydraulics-off mode. The setup included a client program using UI-SimCor with OpenFresco and two server programs using OpenFresco. The client program used the Alpha Operator Splitter integration algorithm with alpha set equal to 1.0, whereby it was equivalent to using the Newmark Explicit integration algorithm. Each server modeled a column of the first story of the structure. To provide baseline test duration and iteration times that all the tests can be compared to, Lehigh University performed a local test involving two PCs on the Lehigh University network. One PC acted as the client and one server while the other PC acted as the second server.

Validation Test Phase 2 involved performing four tests, as summarized in Table 4-3. For Test 2A, Lehigh University acted as the client running UI-SimCor with both servers

running OpenFresco (see Figure 4-6). For Test 2B, Lehigh University acted as the client running UI-SimCor with both servers running OpenFresco (see Figure 4-7). In this test, the same SCRAMNet and target PC setup for Test 1B of Validation Test Phase 1 was utilized except that the stiffness in the actuator model was changed to 1010 kips/in to accommodate the UI-SimCor structure described in Section 3. For Test 2C, University of Illinois at Urbana Champaign (UIUC) acted as the client running UI-SimCor and Lehigh University acted as both servers running OpenFresco, with each server on a separate PC at the RTMD facility (see Figure 4-8).

Test	Client	Server 1	Server 2
2A	Lehigh	Lehigh	Lehigh
2B	Lehigh	Lehigh	Lehigh SCRAMNet
2C	UIUC	Lehigh	Lehigh

Table 4-3 Validation Test Phase 2 test matrix

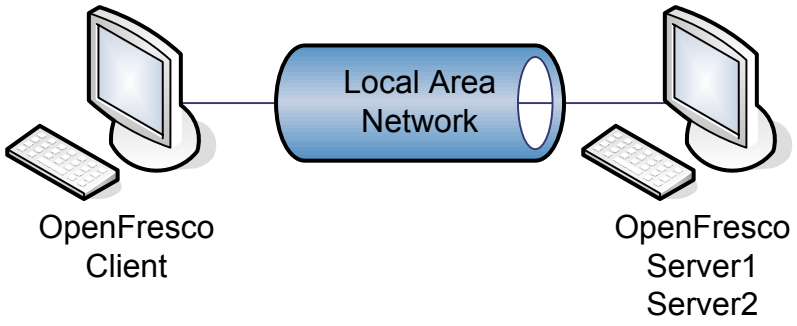


Figure 4-6 Local hybrid simulation setup, Test 2A

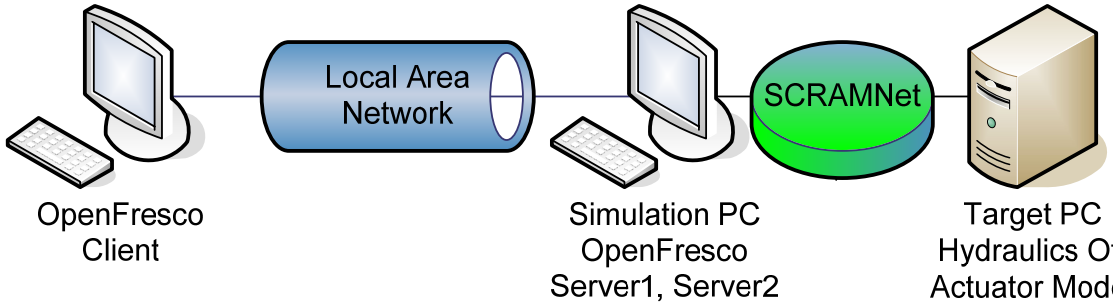


Figure 4-7 Local hybrid simulation setup using SCRAMNet and hydraulics off mode, Test 2B

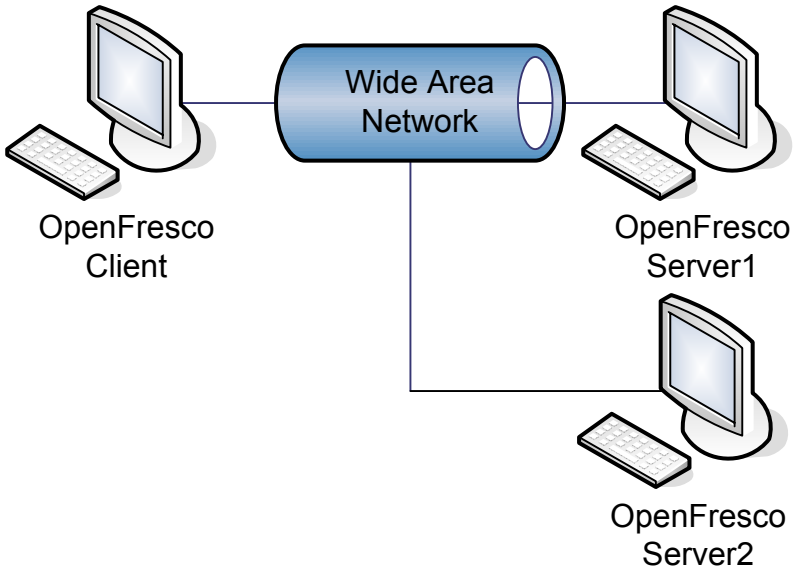


Figure 4-8 Distributed hybrid simulation setup, Tests 2C

The duration of each time step in all tests was recorded along with the total time for the test to be completed. Table 4-4 compares the physical distance between Lehigh University and UIUC along with the average time per time step and total time. The increase in time per time step when compared with the results for the local tests of Validation Test Phase 1 (i.e., Test 1A and 1B) is due to the latency that the MATLAB interface and workspace variable growth adds.

Test	Distance (km)	Average time per time step (ms)	Total time for simulation (min)
2A	0.03	218	14.50
2B	0.03	225	14.83
2C	1100	430	28.78

Table 4-4 Distance and duration comparison for Validation Test Phase 2

A time history analysis of the modified test structure for UI-SimCor had to be performed to check the experimental results against. Therefore, a two pier bridge structure was modeled locally with OpenSees and the displacements at each pier in the horizontal direction were controlled. The time history analysis results were used as a comparison with the results from the tests performed in Validation Test Phase 2. All four experimental results were identical to the time history analysis, as shown below in Figures 4-9 and 4-10.

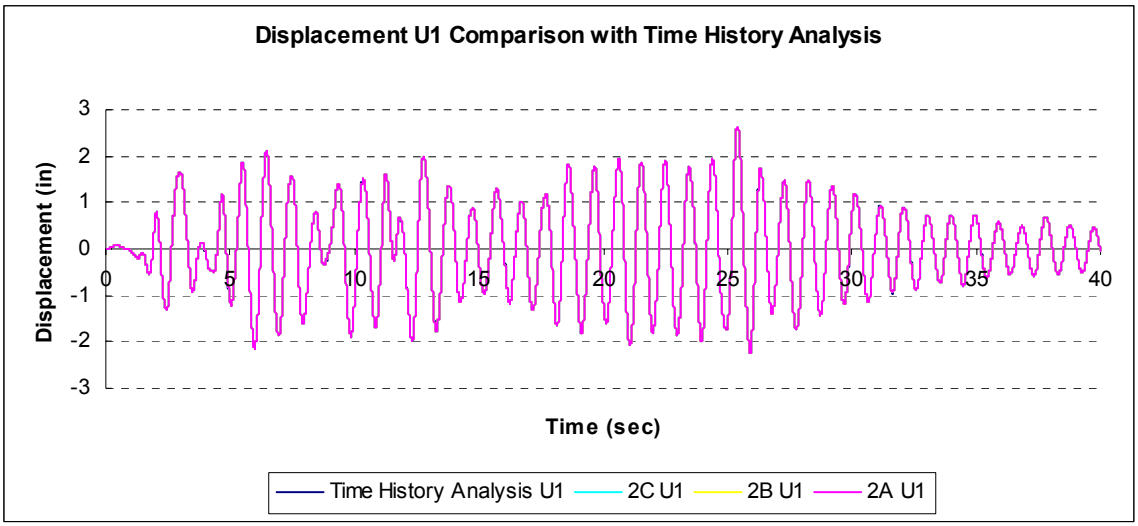


Figure 4-9 Validation Test Phase 2 – comparison of displacement response at DOF U1

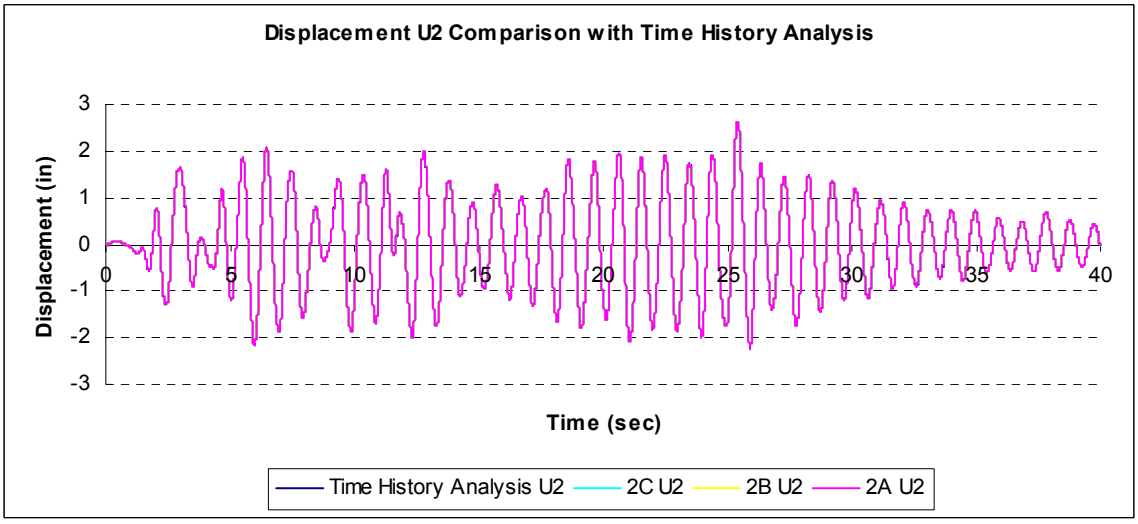


Figure 4-10 Validation Test Phase 2 – comparison of displacement response at DOF U2

4.3 Validation Test Phase 3

Two tests were performed in Validation Test Phase 3. They are summarized in Table 4-5, and include Test 3A and 3B. The tests in Phase 3 are similar to those in Phase 2, except that server 2 controlled an actuator at the RTMD facility. Server 2 communicated with the RTMD Real-time Control Workstation via SCRAMNet, where commands were issued to a free standing actuator and the displacement response was multiplied by a constant equal to the stiffness of one column to produce a simulated force response. Figure 4-11 shows a schematic of the architecture for the communication with RTMD Real-time Control Workstation via SCRAMNet.

In Test 3A, Lehigh University acted as the client running OpenSees with both servers running OpenFresco (see Figure 4-12). In this test, the RTMD Simulation Workstation ran the client and both servers and was running OpenFresco with the SCRAMNet experimental control and MATLAB middleware developed for Validation Test Phase 1.

The RTMD Real-time Control Workstation interacted with the SCRAMNet by reading floating-point commands from the shared memory device and writing floating-point feedback signals back to it based on a predefined memory map. The aforementioned MATLAB program developed in Validation Test Phase 1 gathered the command and feedback signals from the client and RTMD Real-time Control Workstation, respectively. It also converted the command and feedback signals into the required format and placed them in the proper memory locations in SCRAMNet. For this test, two separate trials were run (Test 3A-1 and Test 3A-2) and the details are discussed below. For Test 3B, the same setup was used as in Test 3A, except that UC Berkeley acted as the client (see Figure 4-13).

Test	Client	Server 1	Server 2
3A	Lehigh	Lehigh	Lehigh
3B	UC Berkeley	Lehigh	Lehigh

Table 4-5 Validation Test Phase 3 test matrix

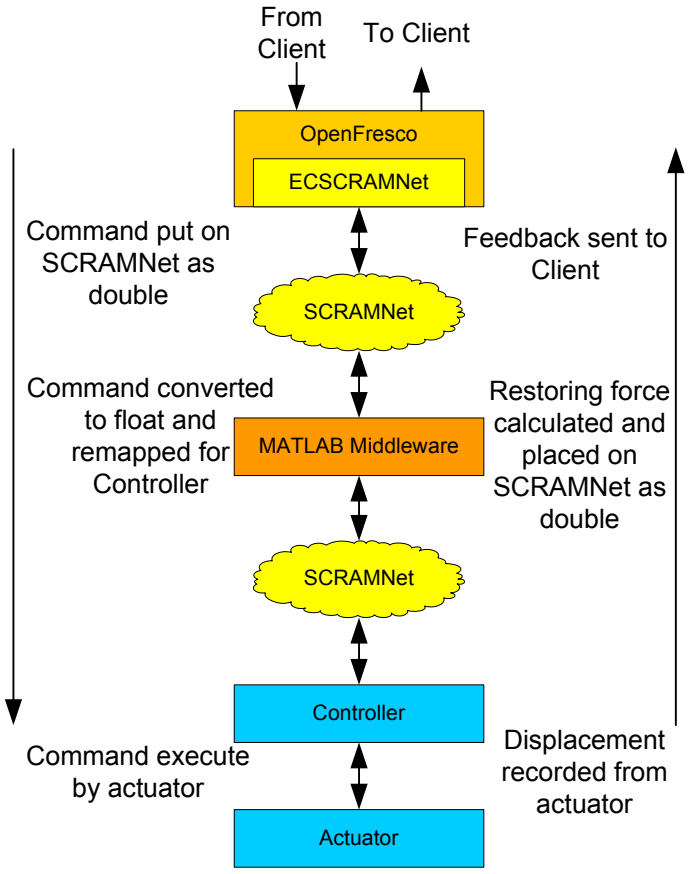


Figure 4-11 Communication with RTMD Real-time Control Workstation via SCRAMNet

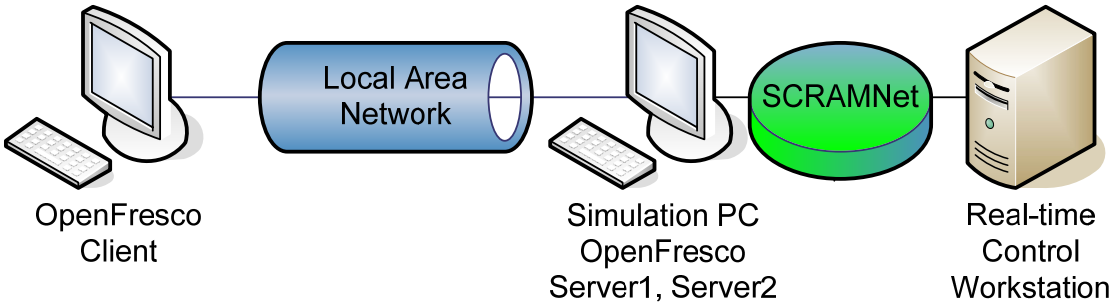


Figure 4-12 Local hybrid simulation setup using SCRAMNet and Real-time hydraulic actuator system, Test 3A

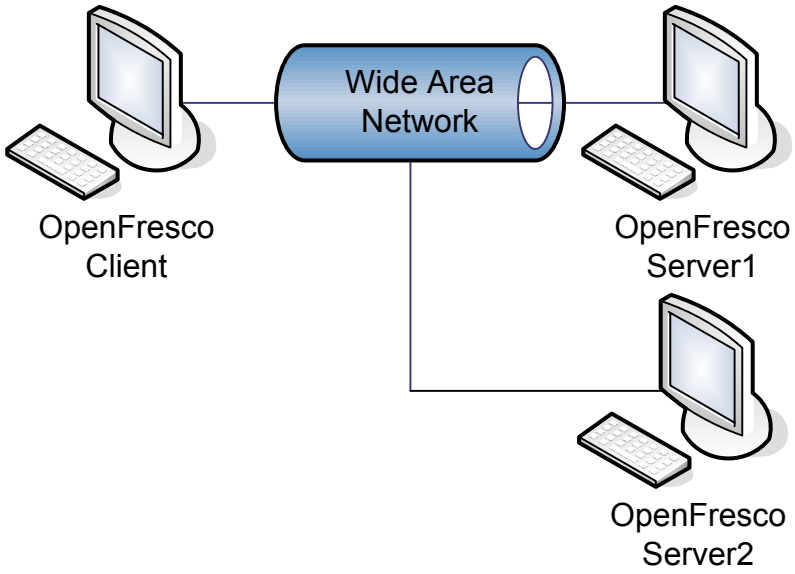


Figure 4-13 Distributed hybrid simulation setup, Tests 2C

The duration of each time step in all tests was recorded along with the total time for each test to be completed. Table 4-6 compares the physical distance between Lehigh University and UC Berkeley along with the average time per time step and total time for the simulations. The time per time step for each of these tests is less than the related tests done in Validation Test Phase 1 since all OpenFresco servers were running on the RTMD Simulation Workstation, eliminating network latency.

.

Test	Distance (km)	Average time per time step (ms)	Total time for simulation (min)
3A-1	0	11	0.74
3A-2	0	31	2.08
3B	4000	392	26.13

Table 4-6 Distance and duration comparison for Validation Test Phase 3

The results for Test 3A-1 differed with respect to amplitude from Test 1A. Shown in Figures 4-14 and 4-15 are a comparison of the displacement history results for DOF displacements from Test 3A-1 and Test 1A. Test 1A is similar to Test 3A-1, except that the former was performed in hydraulics-off mode and therefore had no latency due to the actuator not achieving the command displacement in a specified time. The difference in

the results of Tests 3A-1 and 1A shown in Figures 4-14 and 4-15 is due to the feedback force from the Real-time Target Workstation being read back before the target command displacement was achieved by the actuator. The actuator delay is apparent in Figures 4-16 and 4-17, which show the actuator tracking (actuator command and measured actuator displacement) and synchronization subspace plot of actuator command and measured displacement for Test 3A-1, respectively. The servo-value used on the actuator for Test 3A-1 requires an adequate amount of time for the actuator to achieve its command displacement. By introducing actuator latency, a delay in the restoring force occurs which introduces negative damping into the system. As a result, as can be seen in Figures 4-14 and 4-15 the displacements for Test 3A-1 are larger than that for Test 1A.

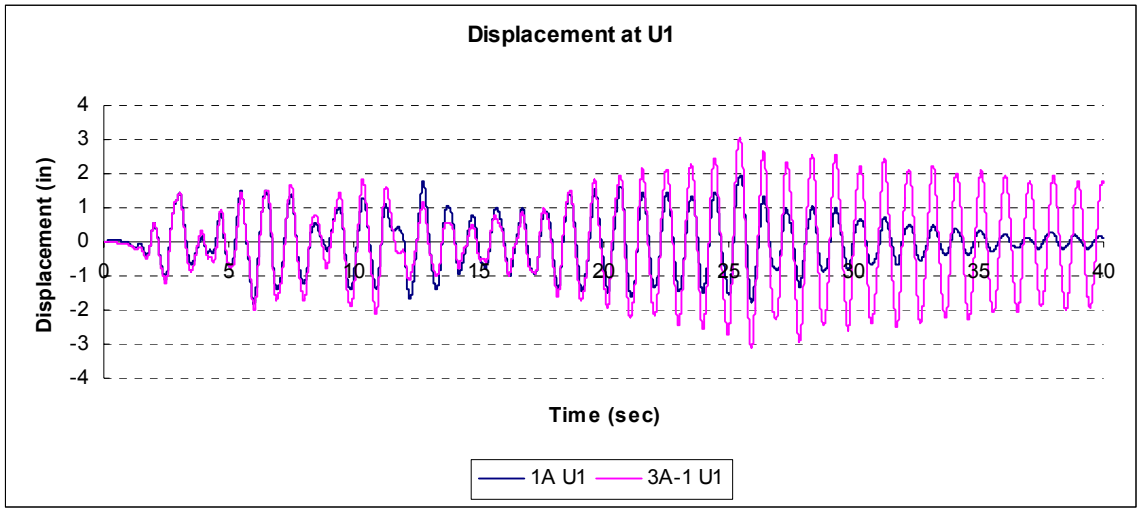


Figure 4-14 Comparison of displacement at U1 for Test 1A and Test 3A-1

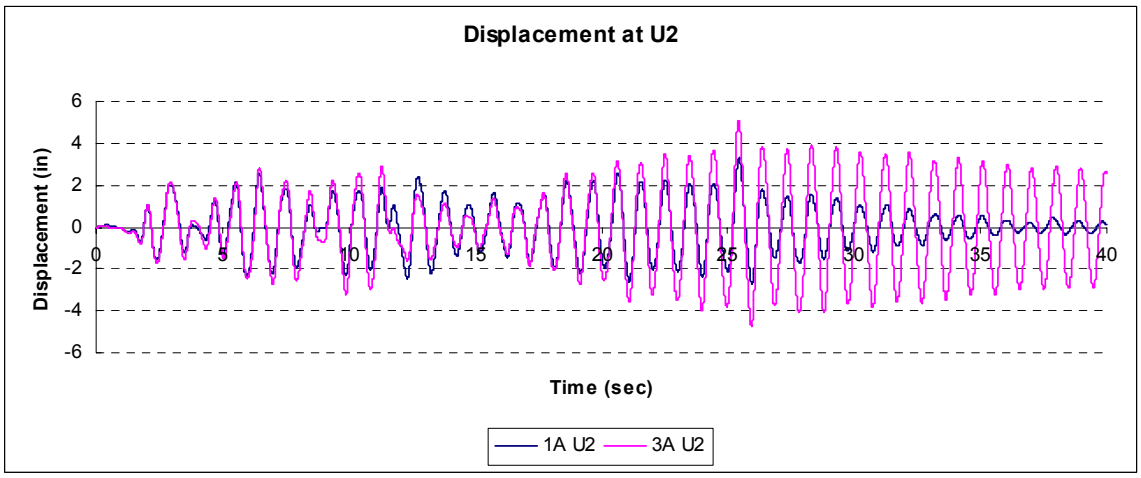


Figure 4-15 Comparison of displacement at U2 for Test 1A and Test 3A-1

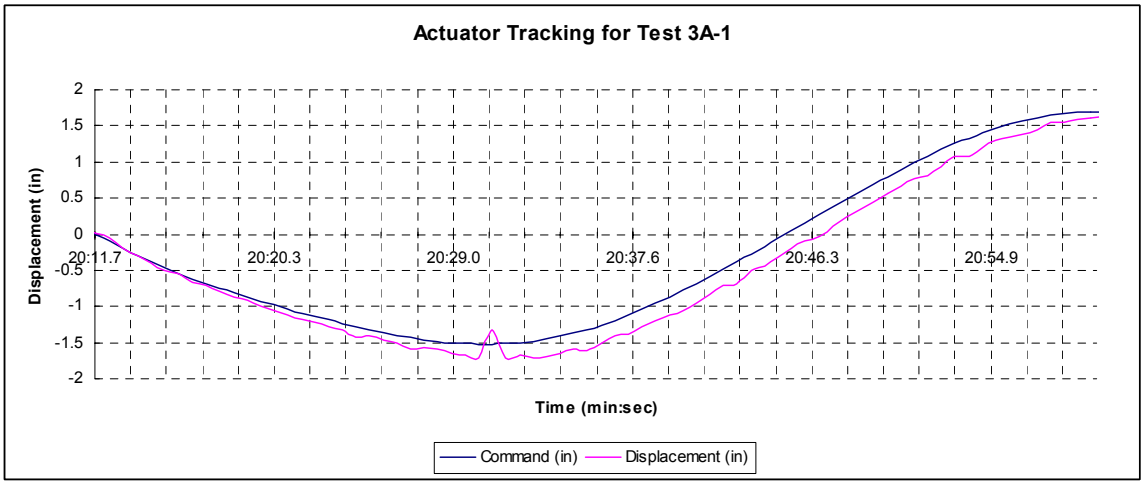


Figure 4-16 Actuator tracking for Test 3A-1

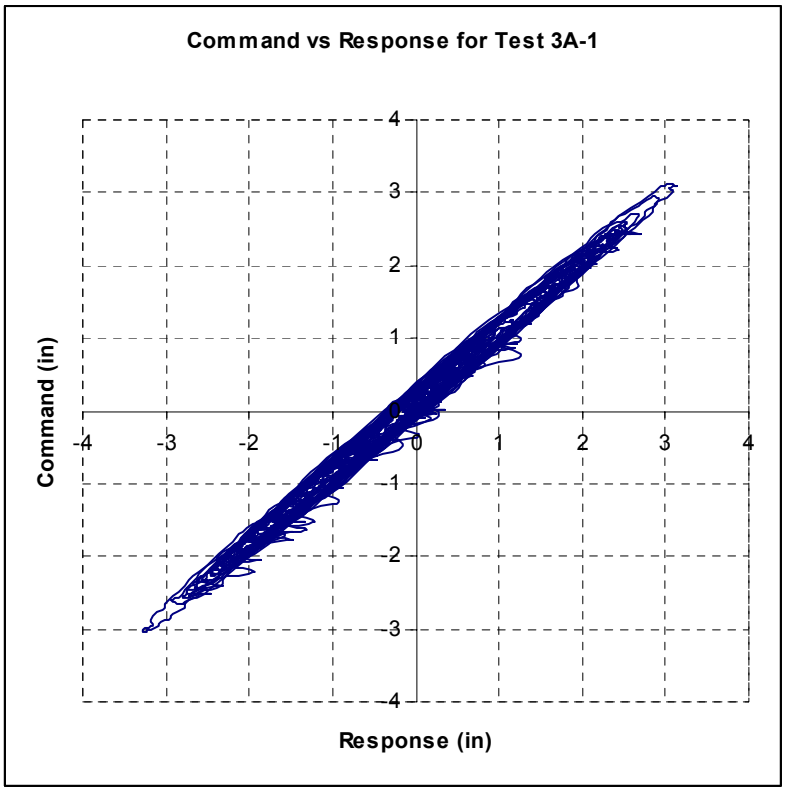


Figure 4-17 Synchronization Subspace Plot of Actuator Command and Measured Displacement for Test 3A-1

The above results show that actuator delay is important and will create errors in the test results if not considered. OpenFresco does not implicitly delay the experimental control modules but does allow for process control through the SCRAMNet experimental control. A memory location is used to alert a listening program via SCRAMNet that OpenFresco has provided a command. The MATLAB middleware program developed for these validation tests listens for this memory change. It then alerts the RTMD Real-time Target PC or Controller to receive the commands from OpenFresco. When the command is achieved, the middleware receives the feedback and alerts OpenFresco via a memory

location change that the response is ready. It is up to the user to allow for the necessary delay in this middleware to avoid actuator delay.

The high speed actuator at the RTMD facility has a delay of about 20ms. The MATLAB middleware program was modified to perform a 20ms delay after imposing the displacement command and a second trial for Test 3A was conducted (named Test 3A-1). It can be seen in the Figures 4-18 through 4-21 that the results for Test 3A-2 are significantly improved over Test3A-1 when the SCRAMNet experimental control in OpenFresco waits until the actuator reaches its command displacement. The results are virtually identical for Test 3B when using the same middleware program for communication as show in Figures 4-20 and 4-21.

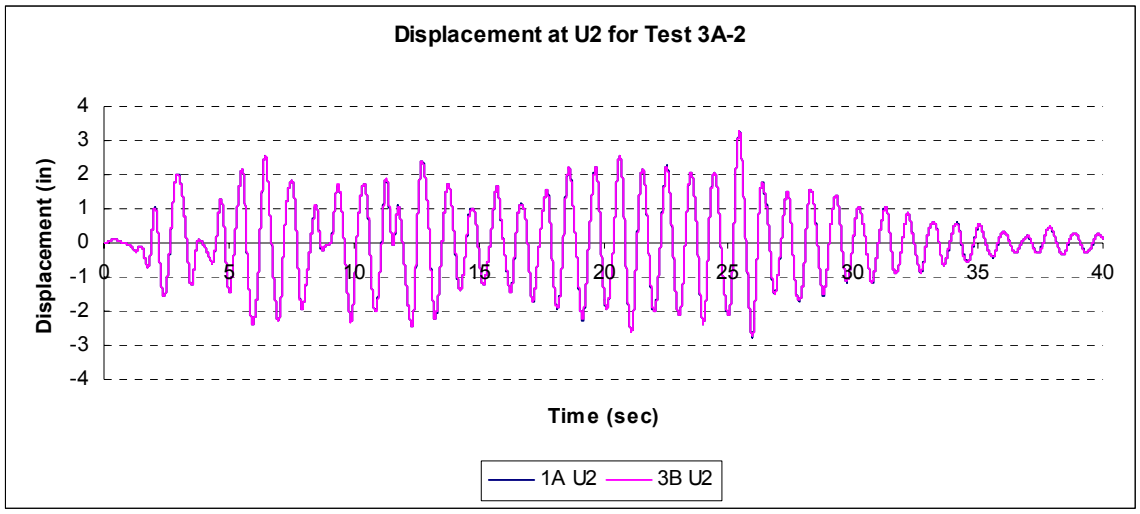


Figure 4-18 Comparison of displacements at U1 for Test 1A and Test 3A-2

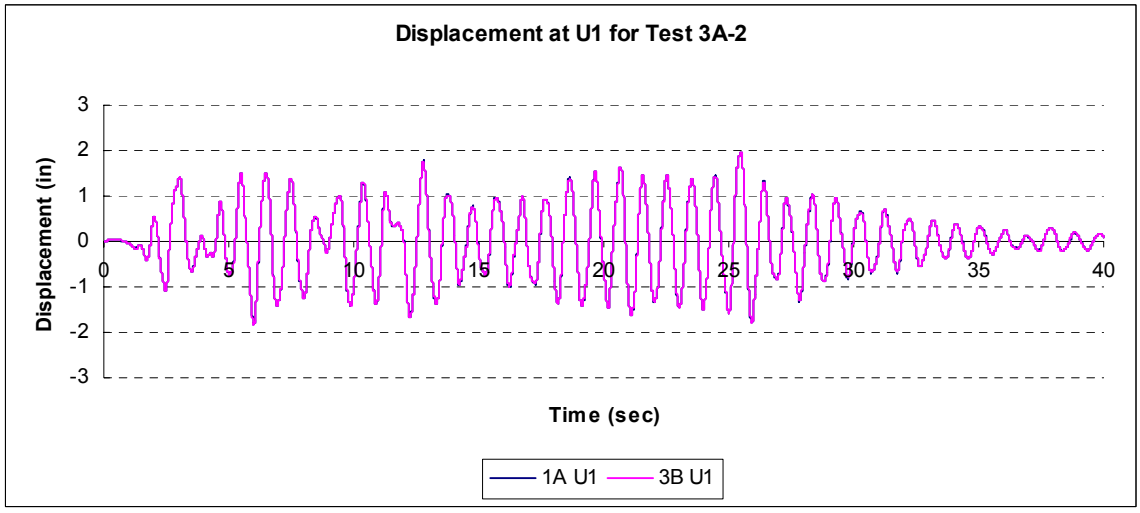


Figure 4-19 Comparison of displacements at U2 for Test 1A and Test 3A-2

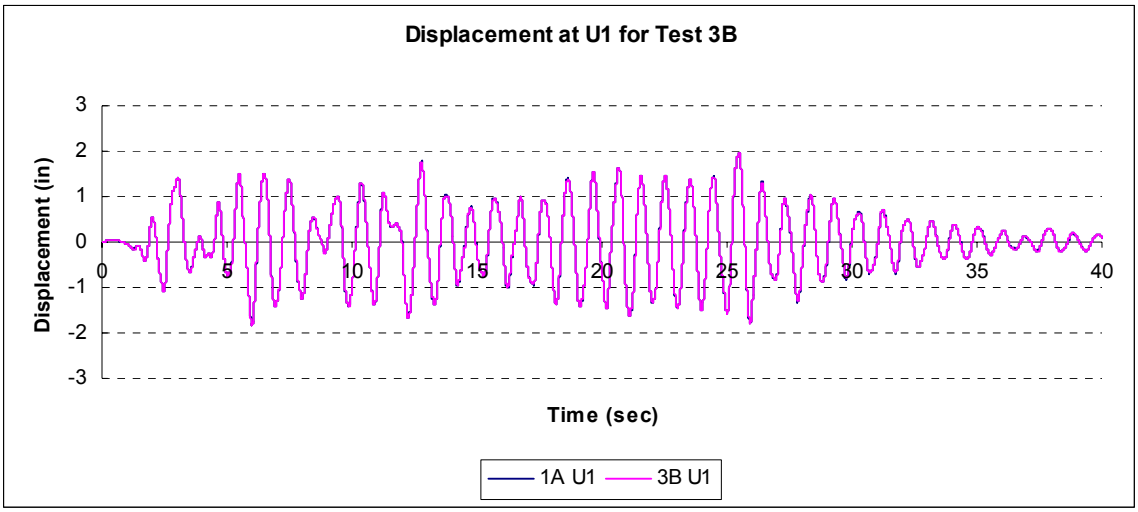


Figure 4-20 Comparison of displacements at U1 for Test 1A and Test 3B

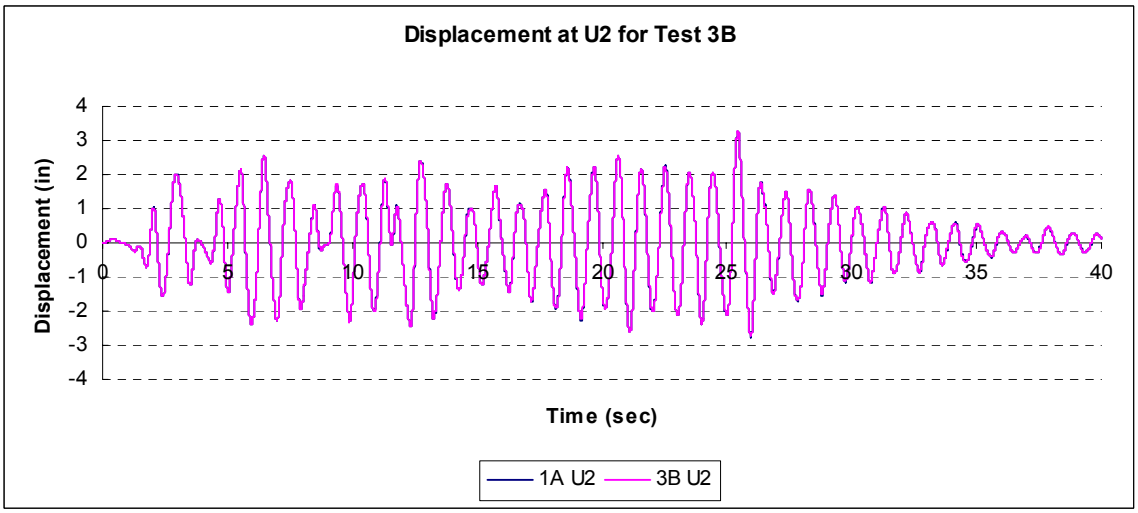


Figure 4-21 Comparison of displacements at U2 for Test 1A and Test 3B

4.4 Validation Test Phase 4

Two tests were performed in Validation Test Phase 4. They are summarized below in Table 4-7, and include Test 4A and 4B. The validation tests in Phase 4 are similar in configuration as the tests in Phase 2, except that in Phase 4, server 2 controlled an actuator at the RTMD facility. Server 2 communicated with the RTMD Real-time Control Workstation via SCRAMNet, where commands were issued to a free standing actuator and the displacement response was multiplied by a constant equal to the stiffness of one column to produce a simulated force response..

For Test 4A, Lehigh University acted as the client running UI-SimCor and both servers running OpenFresco. In this test, the RTMD Simulation Workstation ran the client and both servers and was running OpenFresco with the SCRAMNet experimental control to control the actuator at the RTMD facility (similar to that which was used in Test 3A). For Test 4B, the same setup was used as in Test 4A except UIUC acted as the client

Test	Client	Server 1	Server 2
4A	Lehigh	Lehigh	Lehigh
4B	UIUC	Lehigh	Lehigh

Table 4-7 **Validation Test Phase 4 test matrix**

The duration of each time step in both tests was recorded along with the total time for each test to be completed. Table 4-8 compares the physical distance between Lehigh University and UIUC along with the average time per time step and total time for the simulations.

Test	Distance (km)	Average time per step (ms)	Total time for test (min)
4A	0	245	16.33
4B	1100	474	31.61

Table 4-8 Distance and duration comparison for validation test 4

The displacements of DOF U1 and U2 from Tests 4A and 4B are compared to the results from Test 2A in Figures 4-22 and 4-23, respectively. The displacement at U1 is virtually identical to that from 2A but the displacement at U2, which was controlled by an actuator, shows error buildup towards the end of the record. This is most likely due to the integration method in UI-SimCor not handling the error between displacement command and displacement feedback introduced from the hydraulic actuator system as well as the need to use a reduced time step compared to the integration method used in OpenSees.

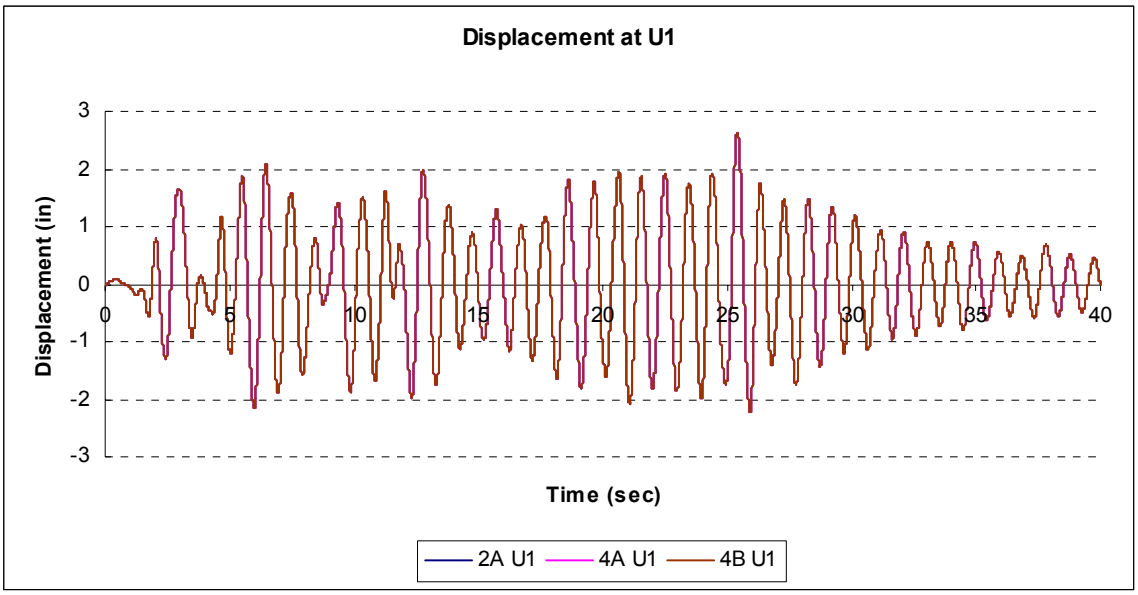


Figure 4-22 Comparison of displacements at U1 for Test 4

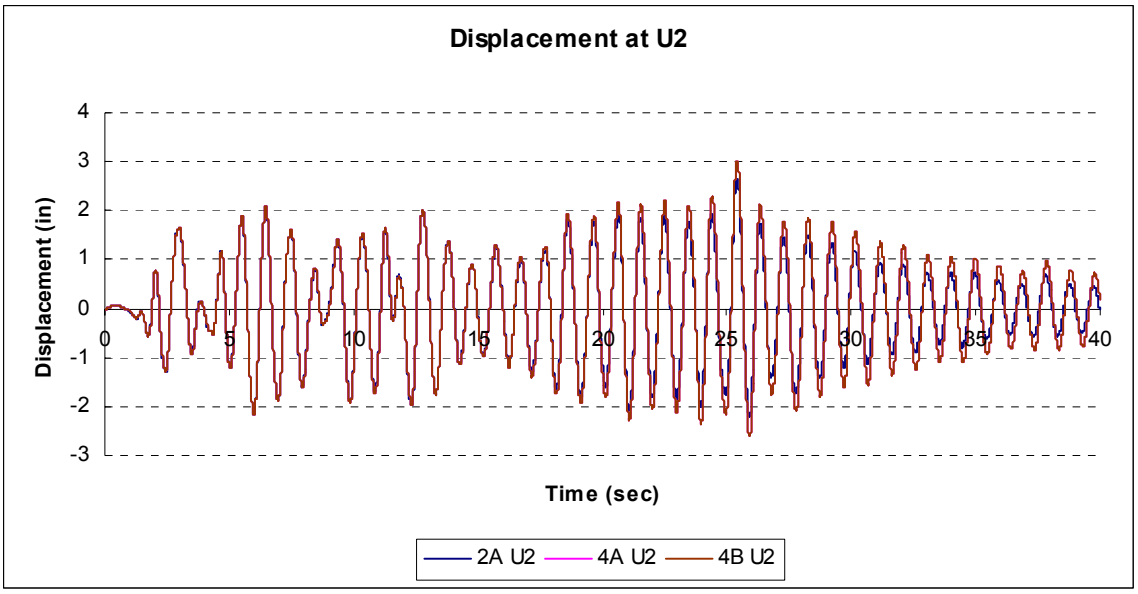


Figure 4-23 Comparison of displacements at U2 for Test 4

4.5 Validation Test Phase 5

The Phase 5 validation test involved performing Test 5A, which was a distributed three site hybrid simulation, where both Lehigh University and University of Colorado at Boulder were using a single actuator setup to produce an experimental response for the client at University of Connecticut. Table 4-9 shows the locations of each component for each test and Figure 4-24 shows the overall system connections. The setup included a client program using OpenSees and OpenFresco version 2.5 and two server programs using OpenFresco version 2.5. The newer version of OpenFresco was used because it was required for the xPC experimental control used at the University of Colorado at Boulder. Server 1 ran at the University of Colorado and communicated with their actuator setup via Target PC and SCRAMNet. The force response was simulated by multiplying the measured displacement response with the elastic stiffness of a column in the structure. Server 2 communicated with the RTMD Real-time Control Workstation via SCRAMNet, where commands were issued to a free standing actuator and the displacement response was multiplied by a constant equal to the stiffness of one column to produce a simulated force response. In this test, a larger and faster hydraulic actuator was used at the RTMD facility to improve performance.

Test	Client	Server 1	Server 2
5A	UConn	Colorado	Lehigh

Table 4-9 Validation Test Phase 5 test matrix

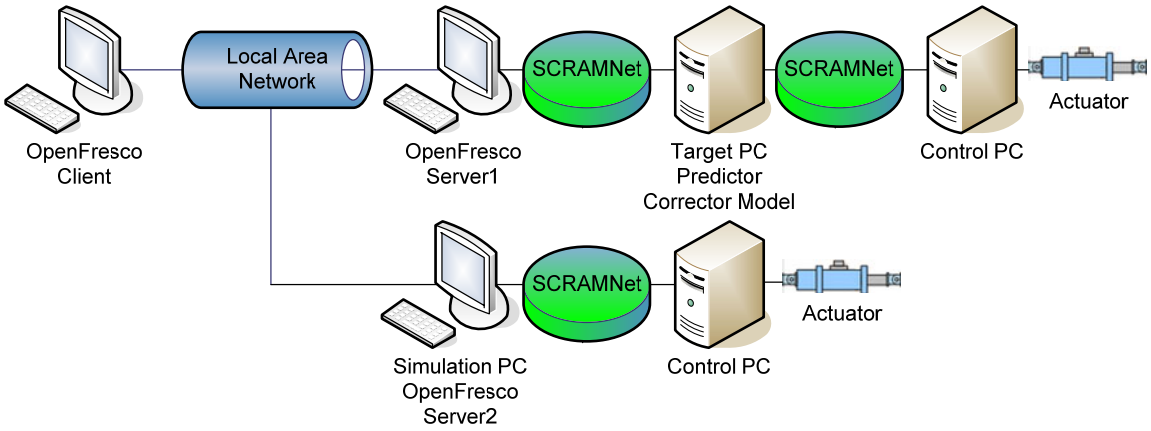


Figure 4-24 Distributed hybrid simulation with dual actuator control for Validation Test Phase 5

For Test 5A, the University of Connecticut acted as the client running OpenSees. Lehigh University utilized the same SCRAMNet actuator control protocol discussed above in Sections 4.3 and 4.4. The University of Colorado utilized the xPCTarget experimental control option in OpenFresco. The xPCTarget experimental control option in OpenFresco utilizes a predictor corrector algorithm to maintain real-time control of the actuator. It obtains new commands from the SCRAMNet memory and updates the memory with feedback data.

The duration of each time step in both tests was recorded along with the total time to complete each of the test. Table 4-10 shown below compares the physical distances between the University of Connecticut and Lehigh University, and the University of Connecticut and the University of Colorado, along with the average time per time step and total time to complete the test. Each time step is done in series, and therefore two round trip transfers have to be made per server for each time step in the test.

Test	Client to Server 1 Distance (km)	Client to Server 2 Distance (km)	Average time per step (ms)	Total time for test (min)
5A	2530	300	181	12.08

Table 4-10 Distance and duration comparison for Validation Test Phase 5

The response from Test 1A was used to compare with the results from Test 5A. Shown below in Figures 4-25 and 4-26 are the displacement histories for DOF U1 and U2, respectively, for Tests 1A and 5A. The results show that the three site hybrid test was successful and matched the results from Test 1A.

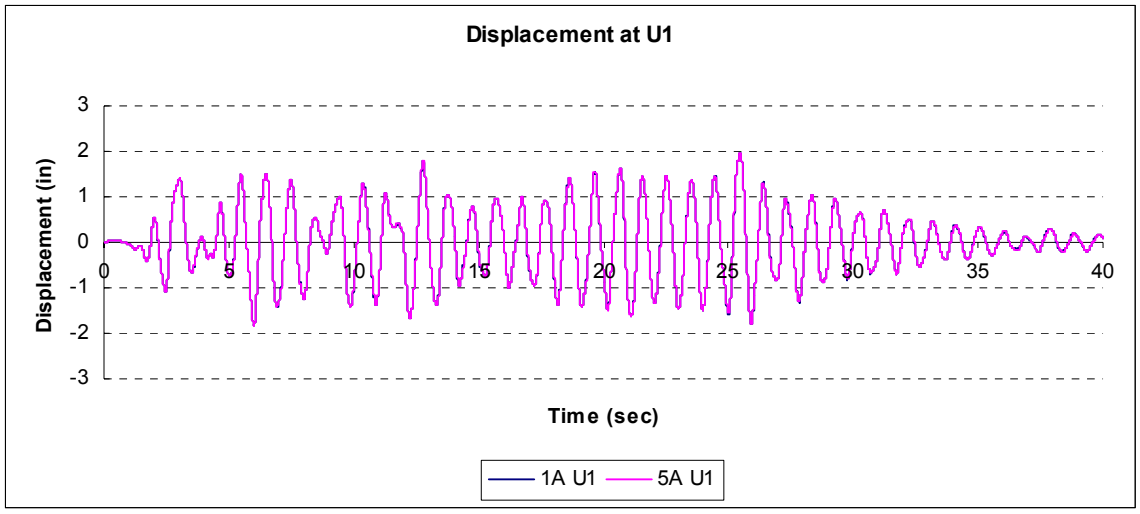


Figure 4-25 Comparison of displacements at U1 for Test 1A and Test 5A

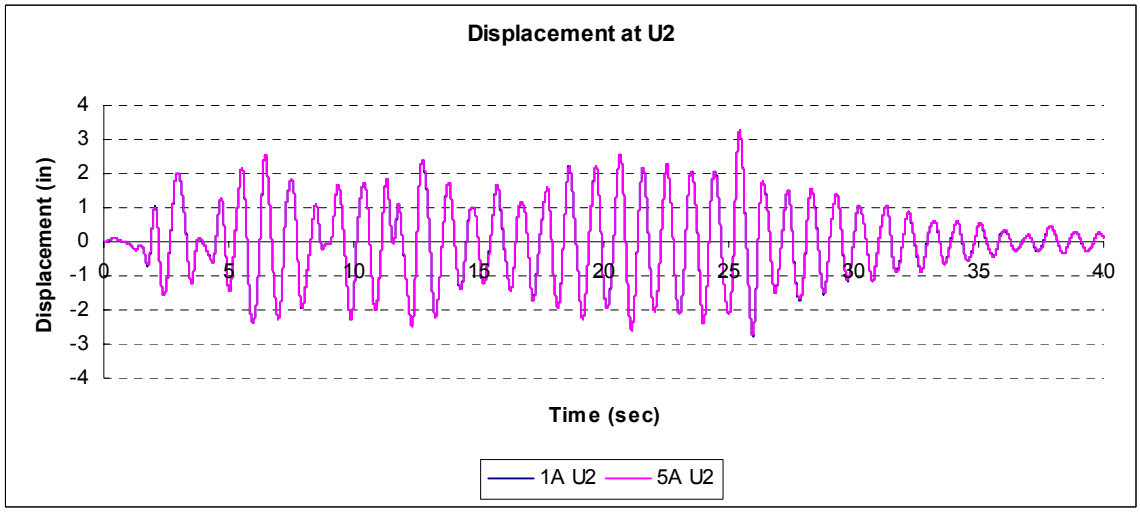


Figure 4-26 Comparison of displacements at U2 for Test 1A and Test 5A

4.6 Validation Test Phase 6

The validation test Phase 6 involved implementing a new integration algorithm into UI-SimCor. Recently, an explicit integration algorithm called the *CR integration algorithm* was developed at Lehigh University (Chen and Ricles, 2008). A pole mapping technique from discrete control theory was utilized to develop the algorithm, where the integration parameters for the algorithm are selected to achieve unconditional stability for a linear elastic structure and nonlinear structures with softening behavior. The CR integration algorithm has the same properties for period elongation and equivalent damping as the Newmark method with constant average acceleration.

Only one validation test (Test 6A) was performed, where the actuators were in hydraulics-off mode. In Test 6A, UI-SimCor was modified to use the unconditionally stable explicit CR integration method. The new integration method replaced the predefined integration algorithm within the UI-SimCor source file, Transient_AlphaOS.m

under the 01_SIMCOR source folder. The same conditions as Test 2A were used to validate the new integration method. The displacement history results from using the CR method for the bridge system used in the validation tests in Phase 2 along with the time history analysis and results from Test 2A are shown in Figures 4-27 and 4-28. These figures show that the CR method produced the same results as the Alpha Operator Splitter (reduced to Newmark Explicit) method used in UI-SimCor.

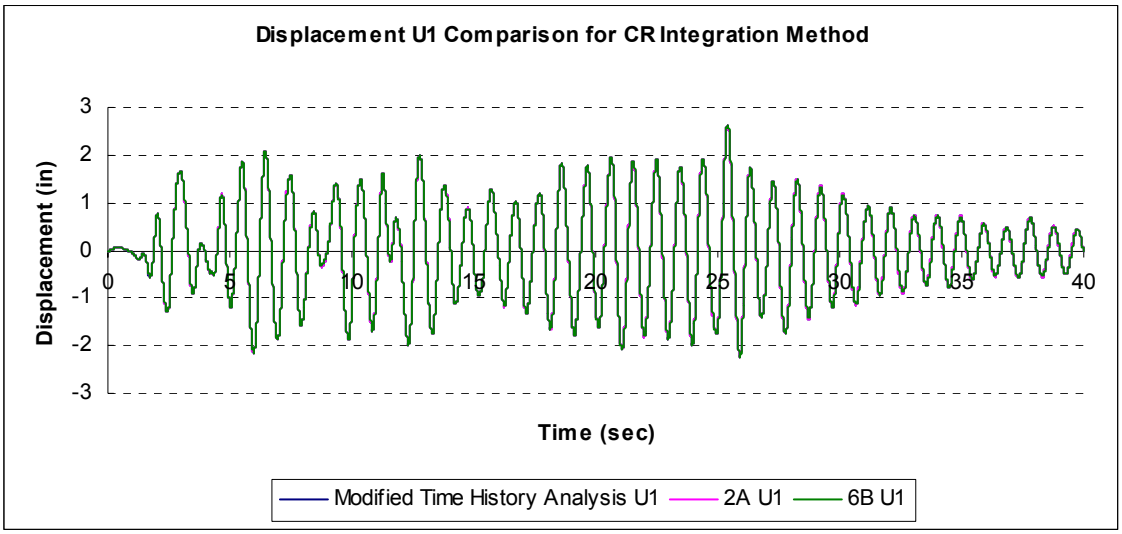


Figure 4-27 Comparison of displacements at DOF U1 for CR algorithm in UI-SimCor

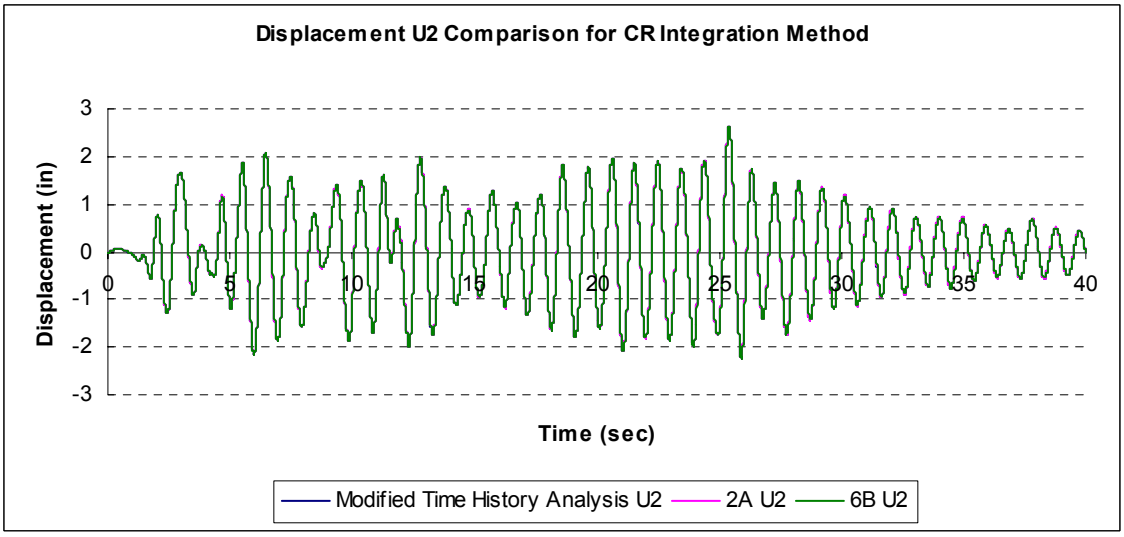


Figure 4-28 Comparison of displacements at DOF U2 for CR algorithm in UI-SimCor

4.7 Validation Test Phase 7

This validation test involved running a modified OpenFresco SCRAMNet experimental control using the RTMD hydraulics-off mode to perform a local real-time hybrid simulation. The setup is shown in Figure 4-29. The OpenFresco client ran on the RTMD Real-time Simulation Workstation which has a SCRAMNet card. The hydraulics-off mode was run on the RTMD Real-time Target Workstation simulating a single actuator force response and used SCRAMNet to communicate commands from and feedbacks to

the RTMD Real-time Simulation Workstation. The sample rate mirrored the 1024Hz sample rate of the Real-time Control Workstation and used the same communication method that the RTMD Real-time Control Workstation utilizes. The OpenFresco SCRAMNet experimental control was modified to simplify the handshaking that the original SCRAMNet experimental control utilized. The modified communications architecture is shown in Figure 4-30. The hydraulics-off mode sets a flag to alert OpenFresco that the SCRAMNet has the latest feedback data and that it can now read SCRAMNet and use the restoring force to generate new actuator displacement commands. OpenFresco reads the flag, resets it, sends commands and waits until the flag is set again.

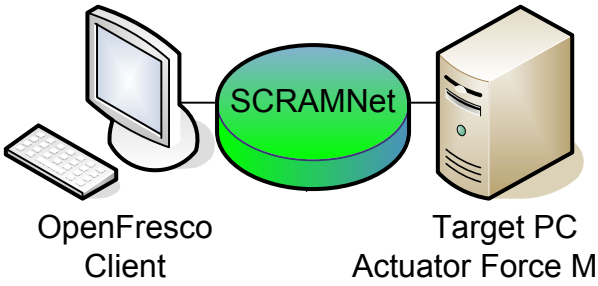


Figure 4-29 Setup for Validation Test Phase 7 involving real-time testing

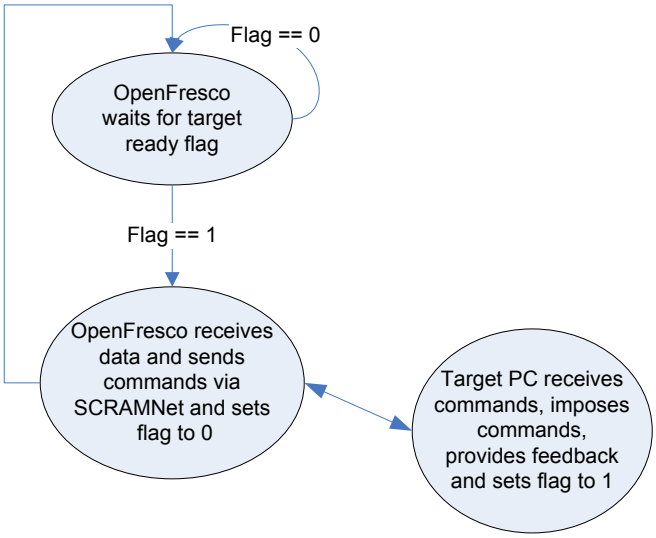


Figure 4-30 Data flow between OpenFresco and Target PC with the modified SCRAMNet experimental control

This method of real-time testing is considered non-deterministic since the OpenFresco client is running under Microsoft Windows XP, which is not a real-time operating system. However, the combination of the SCRAMNet configuration with the speed of the CPU[†] and OpenFresco program allows for this type of testing to work. As noted above, the RTMD Real-time Simulation Workstation was connected to the RTMD Real-time Target Workstation using SCRAMNet for data transfers, where a typical SCRAMNet transfer latency is about 200 nanoseconds, sufficient for real-time testing. The OpenFresco

[†] 2.4 GHz Pentium 4 Xeon CPU

program developed for this test, without the SCRAMNet experimental control, completes one integration in able 0.00025 seconds. When adding the modified SCRAMNet control with the simplified handshaking, it shows that non-deterministic local real-time hybrid testing is possible when using OpenFresco as long as the time per integration step, or Δt, is set to 0.00025 seconds or greater. However, one must account that since Microsoft Windows XP is non-deterministic, there is no guarantee that each time step will be completed in the requested time.

For Test 7A, the only test run in Validation Test Phase 7, the same configuration was used as discussed in the tests of Validation Test Phase 1, except there were no servers and no network communication. The client contained the SCRAMNet experimental control and the RTMD Real-time Target Workstation produced a restoring force response. To see how this real-time configuration responded with respect to different delta T values, four separate trials of this test were conducted where the time per integration step was changed as seen in Table 4-11.

Trial	Time per integration step (sec)	Expected Completion time (sec)
7A-1	0.001	3.905
7A-2	0.005	19.523
7A-3	0.010	39.053
7A-4	0.020	78.105

Table 4-11 Time per integration step for each trial having 4000 total integration steps

To show that each trial was accurate, Figures 4-31 and 4-32 shows that the displacement response of this trial matched the time history analysis performed in Test 1A.

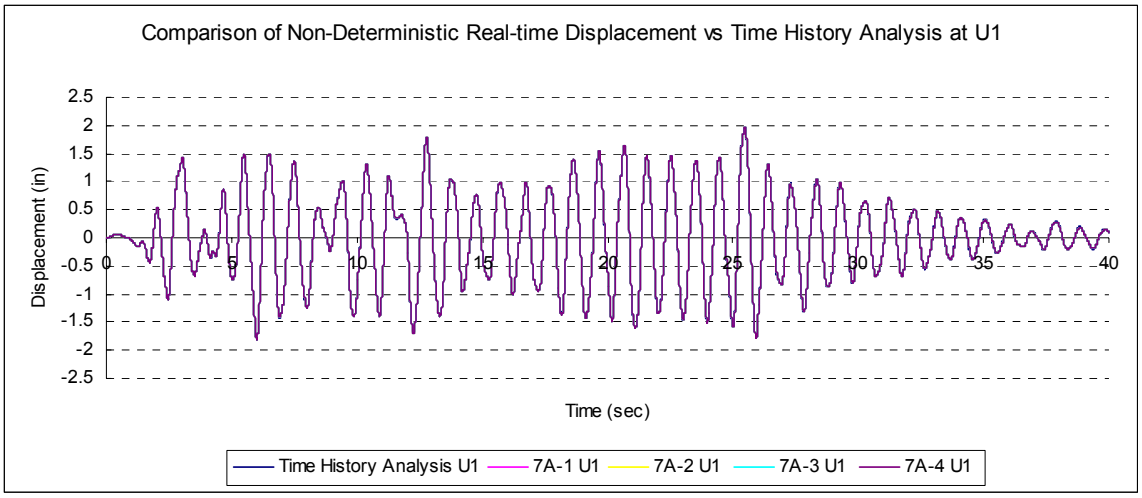


Figure 4-31 Comparison of displacement response of DOF U1 for non-deterministic real-time hybrid simulation

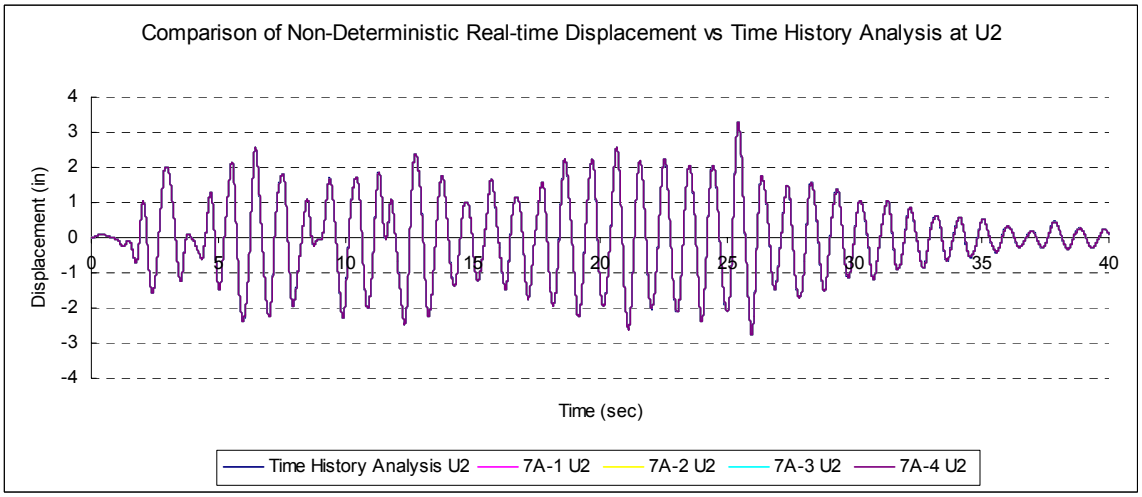


Figure 4-32 Comparison of displacement response of DOF U2 for non-deterministic real-time hybrid simulation

Figure 4-33 shows the elapsed time for each trial compared with the expected elapsed time calculated from the number of time steps and the time per integration step used. With a non-deterministic system, it is impossible to complete in the same elapsed time with each simulation, however the results are acceptable for real-time testing.

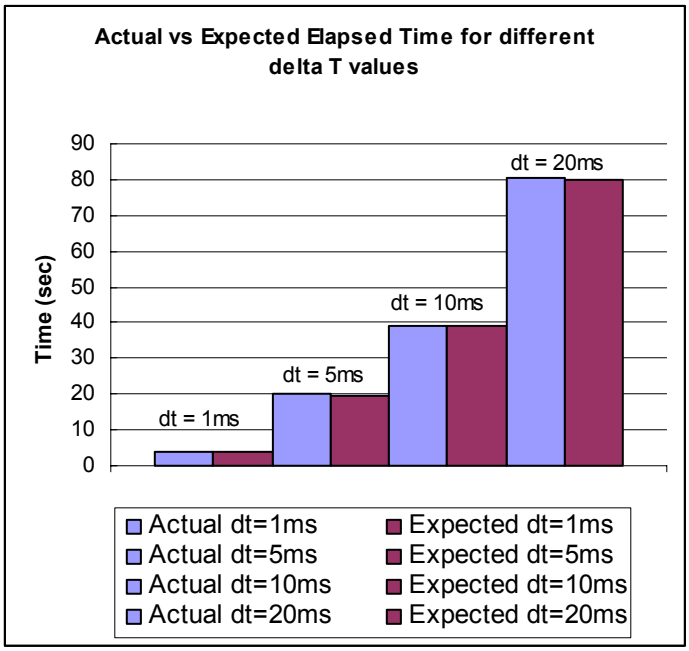


Figure 4-33 Comparison of elapsed time for different delta t values

5 Summary and Recommendations

The NEES RTMD facility at Lehigh University has successfully installed and validated the existing tools for performing hybrid simulations. OpenFresco version 2.0 and UI-SimCor version 2.6 were used to perform numerous local and distributed simulations with other NEES and non-NEES sites. OpenFresco, when used with the OpenSees analysis package, allows for complex and coupled multi-degree-of-freedom systems in both local and distributed configurations. When used with UI-SimCor, only simple, one degree of freedom systems can be used as substructures since the ability to create coupled systems is not currently available. All local and distributed validation tests produced acceptable results and demonstrated that the RTMD facility has successfully implemented the capability of using OpenFresco and UI-SimCor for hybrid simulations.

Validation tests which used hydraulic systems revealed that accounting for actuator delay when converging to a command displacement is important for hybrid simulation. Middleware was developed to bridge OpenFresco and the RTMD facility and it was discovered that when the actuator delay is neglected, the test results are incorrect due to the negative damping that is introduced into the system by the actuator delay. UI-SimCor is more sensitive to actuator delay than OpenSees is. OpenFresco provides a handshaking algorithm for the SCRAMNet experimental control which halts the OpenFresco server until a target flag is set in the memory, enabling the system to wait until the actuator(s) achieve their command displacements. This algorithm needs to be documented in the OpenFresco manual because it was only discovered through studying the source code.

For real-time hybrid simulations, OpenFresco was analyzed and modified to perform a non-deterministic real-time test. UI-SimCor does not have the capabilities of performing

deterministic or fast real-time tests since the average time per iteration is around 200 milliseconds, which is much larger than a typical integration time step of 20 milliseconds. To do deterministic real-time hybrid simulations, the OpenFresco and OpenSees source codes need to be modified, compiled and run on a real-time system or target to ensure reliability. This also may require a customized version of the SCRAMNet experimental control source code depending on system configurations.

When performing distributed hybrid testing, the physical distance between two or more sites is a factor in how long an integration time step takes. In some cases, multiple servers may be running at the same IP or subnet. Currently, OpenFresco handles each remote site separately, and therefore will make N round trip data transfers for N remote sites. To minimize delay when two or more servers have the same IP or subnet, OpenFresco should have a mechanism for packaging the command data to the remote site and performing the packet routing at the remote site through a gateway server that results in only one round trip of data transfer to the same site. Hence, it should also perform the same operation on the feedback sent to the client IP.

UI-SimCor uses the Alpha Operator Splitter integration algorithm for generating control point data. It also provides the Newmark Explicit method. Implementing a new integration method is not clear to the user. In order to implement the integration method, the user must edit the source files. A more effective way of changing and adding integration methods should be available for users of UI-SimCor.

In order to get the SCRAMNet and xPC experimental controls to work with the RTMD facility, a middleware program had to be developed because of the fixed memory structure in OpenFresco. A flexible and configurable memory structure and data types for using SCRAMNet should be available to users of OpenFresco.

6. References

[1] Chang M., Marullo T.M., Ricles J.M., "Hybrid Simulation Testing Method Using OpenFresco", *ATLSS Report,* Lehigh University ATLSS Center, 2007.

[2] Chen C., Ricles J.M., Marullo T.M., "Real-time Hybrid Testing using the Unconditionally Stable Explicit CR Integration Algorithm", *Earthquake Engineering and Structural Dynamics*, submitted for review, 2008.

[3] Chen C., Ricles J.M., "Stability analysis of real-time hybrid testing with actuator delay using explicit integration algorithms", *Earthquake Engineering and Structural Dynamics*, 2007.

[4] Chen C, Ricles JM. "Development of direct integration algorithms for structural dynamics using discrete control theory". *Journal of Engineering Mechanics*; ASCE, Vol. 134(8), pp. 676-683, 2008.

[5] Dermitzakis, S.N. and Mahin, S.A. "Development of Substructuring Techniques for On-Line Computer Controlled Seismic Performance Testing," Report UBC/EERC-85/04, Earthquake Engineering Research Center, University of California, 1985.

[6] Mercan O., Ricles J.M., "NEES @ Lehigh: Real-time Hybrid Pseudodynamic Testing of Large-scale Structures", *Hybrid Simulation: Theory, Implementation and Applications, Chapter 11, Taylor & Francis/ Balkema*, submitted for review, 2007.

[7] Mercan O., Ricles J.M., Sause R., Marullo T.M., "Real-Time Large-Scale Hybrid Testbed for Seismic Performance Evaluation of Smart Structures", *Smart Structures and Systems*, reviewed and accepted for publication, 2007.

[8] MATLAB: The Language of Technical Computing. http://www.mathworks.com/products/matlab/

[9] Simulink: Simulation and Model-Based Design. http://www.mathworks.com/products/simulink/

[10] OpenFresco: Open Framework for Experimental Setup and Control. https://neesforge.nees.org/projects/openfresco/

[11] OpenSees: Software framework for developing applications to simulate the performance of structural and geotechnical systems subjected to earthquakes. http://opensees.berkeley.edu/index.php

[12] UI-SimCor: MATLAB-based simulation coordinator for distributed hybrid simulation and testing. https://neesforge.nees.org/projects/simcor/

[13] Lehigh RTMD Users Guide. http://www.nees.lehigh.edu/index.php/fuseaction/manual.toc, 2007.

Appendix A - Example OpenFresco Client.tcl file

```
# File: Client.tcl
#
# Written: Cheng Chen
# Created: Sept 17 2007
# Revision: A

# ------------------------
# Start of model generation
# ------------------------
# create ModelBuilder (with weo-dimensions and 3 DOF/node)
model BasicBuilder -ndm 2 -ndf 3

# Load OpenFresco package
# ------------------------
# (make sure all dlls are in the same folder as openSees.exe)
loadPackage OpenFresco

# Define geometry for model
# ------------------------
set mass0 1.866
set mass1 5.982
set mass2 4.116

# node $tag $xCrd $yCrd $mass
node  1    0.0    0.0  -mass $mass0 $mass0 0.0
node  2    0.0  144.0  -mass $mass1 $mass1 0.0
node  3    0.0  288.0  -mass $mass2 $mass2 0.0
node  4  360.0    0.0  -mass $mass0 $mass0 0.0
node  5  360.0  144.0  -mass $mass1 $mass1 0.0
node  6  360.0  288.0  -mass $mass2 $mass2 0.0

# set the boundary conditions
# fix $tag $DX $DY $RZ
fix  1  1  1  1
fix  2  0  1  1
fix  3  0  1  1
fix  4  1  1  1
fix  5  0  1  1
fix  6  0  1  1

# Define materials
# ----------------
# uniaxialMaterial Steel02 $matTag $Fy $E $b $R0 $cR1 $cR2 $a1 $a2 $a3 $a4
uniaxialMaterial Elastic 1 505

# Define experimental site
# ------------------------
# expSite RemoteSite $tag <-setup $setupTag> $ipAddr $ipPort <$dataSize>
expSite RemoteSite 1 "192.168.0.10" 8091
expSite RemoteSite 2 "192.168.0.11" 8092

# geometric transformation
# geoTransf type $tag
geomTransf Linear 10

# Define experimental elements
# ----------------------------
# left and right columns

# Define numerical elements
# ------------------------

# Define element

# expElement $eleTag $iNode $jNode $transTag -site $siteTag -initStif $Kij <-iMod> <-rho $rho>
expElement beamColumn 1 1 2 10 -site 1 -initStif 18407 0 0 0 505 -36334 0 -36334 3488056

# element elasticBeamColumn $eleTag $iNode $jNode $A $E $Iz $transTag
element elasticBeamColumn 2 2 3 91.4 29000 4330 10
```

```
# expElement $eleTag $iNode $jNode $transTag -site $siteTag -initStif $Kij <-iMod> <-rho $rho>
expElement beamColumn 3 4 5 10 -site 2 -initStif 18407 0 0 0 505 -36334 0 -36334 3488056

element elasticBeamColumn 4 5 6 91.4 29000 4330 10
element elasticBeamColumn 5 2 5 44.2 29000 9040 10
element elasticBeamColumn 6 3 6 44.2 29000 9040 10


# Define dynamic loads
# --------------------
# set time series to be passed to uniform excitation
set dt 0.01
set scale 1.0
set accelSeries "Path -filePath ELC270.txt -dt $dt -factor [expr 386.1*$scale]"

# Get Initial Stiffness
# ---------------------
#initialize

# create UniformExcitation load pattern
# pattern UniformExcitation $tag $dir
pattern UniformExcitation 11 1 -accel $accelSeries

# Define damping
# rayleigh $alphaM $betaK $betaKinit $betaKcomm
rayleigh  0.18289   0   0.0017984   0

# ------------------------
# End of model generation
# ------------------------

# ----------------------------
# Start of recorder generation
# ----------------------------
# create the recorder objects
recorder Node -file Node_Dsp.out -time -node 1 2 3 4 5 6 -dof 1 disp
recorder Node -file Node_Vel.out -time -node 1 2 3 4 5 6 -dof 1 vel
recorder Node -file Node_Acc.out -time -node 1 2 3 4 5 6 -dof 1 accel
recorder Element -file Elmt_Frc.out -time -ele 1 2 3 4 5 6 force
recorder Element -file Elmt_Def.out -time -ele 1 2 3 4 5 6 deformation
# --------------------------
# End of recorder generation
# --------------------------

# ----------------------------
# Start of analysis generation
# ----------------------------
# create the system of equations
system BandGeneral

# create the DOF numberer
numberer Plain
# create the constraint handler
constraints Plain
# create the convergence test
test EnergyIncr 1.0e-6 10

# create the integration scheme
integrator NewmarkExplicit 0.5

# create the integration algorithm
algorithm Linear

# create the analysis object
analysis Transient
# --------------------------
# End of analysis generation
# --------------------------

# ----------------------------
# Finally perform the analysis
# ----------------------------
```

```
set pi 3.14159265358979
set lambda [eigen 2]
puts "\nEigenvalues at start of transient:"
puts "   lambda\t   omega\t   period"
foreach lambda $lambda {
        set omega [expr pow($lambda,0.5)]
        set period [expr 2*$pi/pow($lambda,0.5)]
        puts "$lambda  $omega  $period"
}

# open output file for writing
set outFileID [open elapseTime.txt w]
# perform the transient analysis
set tTot [time {
        for {set i 1} {$i<4000} {incr i} {
                set t [time {analyze 1 $dt}]
                puts $outFileID $t
        }
}]
puts "Elapsed Time = $tTot \n"

# close the output file
close $outFileID

wipe
# ---------------
# End of analysis
# ---------------
```

Appendix B - Example OpenFresco Server.tcl file for OpenSees Control

```
# File: Server1.tcl (use with Client.tcl)
#
# Written: Cheng Chen
# Created: Sept 17 2007
# Revision: A

# --------------------------------
# Start of model generation
# --------------------------------
# create ModelBuilder (with two-dimensions and 3 DOF/node)
model BasicBuilder -ndm 2 -ndf 3

# Define materials
# ----------------
# uniaxialMaterial Elastic $matTag $E
uniaxialMaterial Elastic 1 505

# Define experimental control
# ---------------------------
# expControl SimUniaxialMaterials $tag $matTags
expControl SCRAMNet 2 0 1

# Define experimental setup
# -------------------------
# expSetup OneActuator $tag <-control $ctrlTag> $dir <-ctrlDispFact $f> ...
expSetup OneActuator 1 -control 1 2

# Define experimental site
# ------------------------
# expSite LocalSite $tag $setupTag
expSite ActorSite 1 -setup 1 8091

# --------------------------------
# End of model generation
# --------------------------------

# --------------------------------
# Start the server process
# --------------------------------
# startSimAppSiteServer $siteTag $port
startLabServer 1

# ---------------------------------
# End of analysis
# ---------------------------------
```

Appendix C – Example UI-SimCor SimConfig.m File

```
function [Sys, MDL, AUX] = SimConfig
MDL = MDL_RF; AUX = MDL_AUX;        % Type definition. Do not delete this line.

% SimConfig.m
% _____________________________________________________________________________
%
% Common parameters
% _____________________________________________________________________________

% Ground acceleration file name with extension. The file should contains two
% columns for time and acceleration. The unit of acceleration should be
% consistent with the mass, time, and force. (i.e. mass*acc = force)
Sys.GM_Input = 'elcentro.dat';

% Ground acceleration scale factor. This factor will be multiplied to
% acceleration before starting simulation ( 386.1 in/s^2, or 9810mm/s^2).
Sys.GM_SC = 386.1;

% Direction of ground acceleration. (x, y, or z)
Sys.GM_direction = 'x';

% Integration parameter for explicit newmark method

Sys.Alph = 0.0;
Sys.Beta = 0.0;
Sys.Gamm = 1/2 + Sys.Alph;

% Evaluate Stiffness?
% Yes (1) to run stiffness evaluation test,
% No  (0) to read stiffness matrix from file. In this case, there should exist
%         stiffness matrices of individual module in the files MDL01_K.txt,
%         MDL02_K.txt, etc.
Sys.Eval_Stiffness = 1;

% Number of initial static loading steps. When there exist static constant
% loading,i.e. gravity forces, apply then in Zeus-NL or OpenSees as a
% incremental loading with 'n' steps. In this file, SimConfig.m, specify the
% number of static steps in the following variable.
Sys.Num_Static_Step = 0;

% Number of dynamic analysis steps
Sys.Num_Dynamic_Step = 4000;

% Dynamic analysis time steps
Sys.dt = 0.01;

% Rayleigh damping, xi_1 and xi_2: Damping ratio, Tn_1, Tn_2: Target period
Sys.xi_1 = 0.02;
Sys.Tn_1 = 0.976;
Sys.xi_2 = 0.02;
Sys.Tn_2 = 0.397;

% Number of Stiffness test
% If stiffness is evaluated through experiment, the evaluation need to be done
% several times and the average of the results are used as the initial
% stiffness. This parameter is used when Sys.Eval_Stiffness = 1
Sys.Num_Test_Stiffness  = 1;

% Enable GUI for SimCor?
% Yes (1) enable the GUI for SimCor
% No  (0) disable the GUI for SimCor
%         Hybrid simulation will be run automatically.
%         Not recommended for the experiment.
Sys.EnableGUI       = 1;        % Use GUI for SimCor

% Number of restoring force modules.
Sys.Num_RF_Module     = 2;
```

```
% Number of auxilary modules.
Sys.Num_AUX_Module    = 0;

% Total number of effective nodes. Effective nodes are interface nodes between
% modules and nodes where lumped masses are defined.
Sys.Num_Node       = 2;

% Lumped mass assigned for each DOF for each node.
% Node number = x, y, z, rx, ry, rz directional mass
Sys.Node_Mass{1} = [23.928, 0, 0, 0, 0, 0];
Sys.Node_Mass{2} = [23.928, 0, 0, 0, 0, 0];

% ____________________________________________________________________________________________
%
% Restoring force module configuration
% ____________________________________________________________________________________________

% Create objects of MDL_RF
MDL(1) = MDL_RF;
MDL(2) = MDL_RF;
% Name of each module.
MDL(1).name = 'first floor'; % Module ID of this module is 1
MDL(2).name = 'second floor'; % Module ID of this module is 2

% URL of each module
% Format - IP address:port number
MDL(1).URL  = '192.168.0.10:8090';
MDL(2).URL  = '192.168.0.11:8091';


% Communication protocol for each module.
%      NTCP        : communicate through NEESPOP server
%      TCPIP       : binary communication using TCPIP
%      LabView1    : ASCII communication with LabView plugin format (Propose-Query-Execute-Query)
%      LabView2    : same as LabView1 but Propose-Query
%      OpenFresco1D : OpenFresco, only 1 DOF is implemented now.
%      NHCP        : NHCP, linear 1 DOF simulation mode, Mini MOST 1 and 2 at UIUC or SDSC
MDL(1).protocol = 'OpenFresco1D';
MDL(2).protocol = 'OpenFresco1D';

% Module 1:  ------------------------------------------------------------------------------
MDL(1).node    = [1];          % Control point node number
MDL(1).EFF_DOF = [1 0 0 0 0 0]; % Effective DOF for CP 1

MDL(2).node    = [2];          % Control point node number
MDL(2).EFF_DOF = [1 0 0 0 0 0]; % Effective DOF for CP 2


% Displacement for preliminary test for each module
% Del_t: Translation, Del_r: Rotation in radian
MDL(1).DEL_t = 0.005;
MDL(1).DEL_r = 0.002;
MDL(2).DEL_t = 0.005;
MDL(2).DEL_r = 0.002;

% Enable GUI for each module?
% GUI for each module can only display the data.
% GUI for each module can not control the hybrid simulation.
% Yes (1) enable the GUI for each module
% No  (0) disable the GUI for each module
MDL(1).EnableGUI = 0;
MDL(2).EnableGUI = 0;

% ____________________________________________________________________________
%
% Advanced modular parameters
% ____________________________________________________________________________
% These parameters need to be redefined for following situations.
%    (1) Different coordinate system between UI-SIMCOR and static module
%    (2) When scale factor needs to be applied either in experiment or
%        simulation
%    (3) To define force and displacement criteria (for tolerance and safety)
%    (4) To trigger camera modules or DAQ system
```

```
%    (5) When LBCB at UIUC is used for experiment
%    (6) When NHCP protocol is used
%
% URL of remote site and NHCP mode for NHCP
for i=1:Sys.Num_RF_Module
  if strcmp(lower(MDL(i).protocol), 'nhcp')
    MDL(i).remote_URL = '127.0.0.1:99999';
    MDL(i).NHCPMode = 'sim1d';
  end
end

% Stiffness for NHCP (Only valid if NHCPMode = 'Sim1D')
for i=1:Sys.Num_RF_Module
  if strcmp(lower(MDL(i).NHCPMode), 'sim1d')
    MDL(i).NHCPSimK = '1000';
  end
end

% Coordinate transformation. If it needs, the transformation matrix also
% needs to be provided.
for i=1:Sys.Num_RF_Module
  MDL(i).TransM = [];
end

% Scale factor for displacement, rotation, force, moment
% Experimental specimens are not always in full scale. Use this factors to
% apply scale factors.
% The displacement scale factors are multiplied before they are
% sent to module. Measured force and moments are divided with scale factors
% before used in the PSD algorithm.
for i=1:Sys.Num_RF_Module
  MDL(i).ScaleF = [1 1 1 1];  % Module i
end

% Relaxation check
% If this parameter is 1, UI_SimCor send commend to retrieve data and check
% relaxation just before the execution of proposed command. If it's 1, the
% checking criteria needs to be provided.
for i=1:Sys.Num_RF_Module
  MDL(i).CheckRelax  = 0;   % Module i
  % if MDL(i).CheckLimit=1, define following variables.
  % Variable size should be (number of control nodes)* 6 array
  %
  % Displacement variation ratio (not increment)
  % MDL(i).MES_D_inc = [ a b c d e f
    %                    ...       ];
  % Force variaiton ratio (not increment)
  % MDL(i).MES_F_inc = [ a b c d e f
    %                    ...       ];
end

% Check displacement and force limit
% At every steps, check if the displacement or force are approaching to the
% limitation of the equipments stroke or force capacity.
for i=1:Sys.Num_RF_Module
  MDL(i).CheckLimit  = 0;   % Module i
  % if MDL(i).CheckLimit=1, define following variables.
  % Variable size should be (number of control nodes)* 6 array
  %
  % Displacement increment limit(not ratio)
  % MDL(i).TGT_D_inc = [ a b c d e f
    %                    ...       ];
    % Displacement limit
  % MDL(i).CAP_D_tot = [ a b c d e f
    %                    ...       ];
    % Force limit
  % MDL(i).CAP_F_tot = [ a b c d e f
    %                    ...       ];
    % Displacement tolerance (ratio)
  % MDL(i).TOL_D_inc = [ a b c d e f
    %                    ...       ];
end
```

```
% Loading and Boundary Condition Box (LBCB) case. If it's 1, the
% coordinate transformation matrix needs to be provided.
% This can be also used for any other actuator which has diffrence number of
% DOF coordinate with those of UI-SIMCOR

for i=1:Sys.Num_RF_Module
  MDL(i).LBCB = 1;
end

for i=1:Sys.Num_RF_Module
  MDL(i).LBCB_TransM = [0 1 0;-1 0 0];
end

% _____________________________________________________________________________________________
%
% Auxiliary module configuration
% _____________________________________________________________________________________________

% AUX(1)                = MDL_AUX;
% AUX(1).URL            = '127.0.0.1:12000';
% AUX(1).protocol       = 'labview1';
% AUX(1).name           = 'Camera';     % Module ID of this mdoule is 1
% AUX(1).Command        = {'displacement' 'z' 3500};
```

Appendix D - Example OpenFresco Server.tcl file for UI-SimCor Control

```
# File: Server2.tcl
#
# Equivalent Second Floor( kips, in, s)
#
# $Revision: $
# $Date: $
# $URL: $
#
# Written: Jun Cao (juc3@lehigh.edu)
# Created: Sep. 18, 2007
# Revision: A

# --------------------------------
# Start of model generation
# --------------------------------
# create ModelBuilder (with two-dimensions and 2 DOF/node)
model BasicBuilder -ndm 2 -ndf 3

# Define geometry for model( equivalent to first floor)
# -------------------------
# node $tag $xCrd $yCrd $mass
node  2     0.0  0.0
node  3     0.0  144.0

# Define materials
# ----------------
# uniaxialMaterial Steel02 $matTag $k
uniaxialMaterial Elastic 1 1010

# Define experimental control
# ---------------------------
# expControl SimUniaxialMaterials $tag $matTags
expControl SCRAMNet 2 0 1

# Define experimental setup
# -------------------------
# expSetup OneActuator $tag <-control $ctrlTag> $dir <-ctrlDispFact $f> ...
expSetup OneActuator 2 -control 2 1

# Define experimental site
# ------------------------
# expSite ActorSite $tag -setup $setupTag $ipPort <$dataSize>
#expSite ActorSite 2 -setup 2 8091
expSite LocalSite 2 2

#geometric transformation
#geomTransf type $tag
geomTransf Linear 10

# Define experimental element
# ---------------------------
# equivalent to first floor
# expElement zeroLength $eleTag $iNode $jNode -dir $dirs -site $siteTag -initStif $Kij <-orient $x1 $x2 $x3 $y1 $y2 $y3> <-iMod>
<-mass $m>

expElement zeroLength 2 2 3 -dir 2 -site 2 -initStif 1010  -orient 0 1 0 -1 0 0

# --------------------------------
# End of model generation
# --------------------------------

# --------------------------------
# Start the server process
# --------------------------------

# startSimAppElemServer $eleTag $port
startSimAppSiteServer 2 8092
```

```
# ---------------------------------
# End of analysis
# ---------------------------------
```

Appendix E - MATLAB Middleware File for SCRAMNet Communication with RTMD Equipment

```
%% Create SCRAMNet Object
scr = edu.lehigh.nees.scramnet.ScramNetIO;
scr.initScramnet;

%% SCRAMNet Addresses for control bits
D_AVAILABLE = 500;
D_COMMAND = 501;
D_COMMAND_TIME = 505;
D_STATE = 506;
D_DISP_FEEDBACK = 507;
D_FORCE_FEEDBACK = 510;
D_FEEDBACK_TIME = 511;
F_PAUSE_BIT = 62;
F_SIM_BIT = 0;
F_COMMAND = 1;
F_DISP_FEEDBACK = 67;

%% SCRAMNet constants for control
PROCESSING = 1;
DONE = 0;

%% Simulation Parameters
SCALEFACTOR = 1; % Just assume mm in this case
LOADSCALE = 505;
UPPERLIMIT = 10; %mm
LOWERLIMIT = -10; % mm

%% Create CSV Writer to log data
csv = edu.lehigh.nees.util.CSVWriter;
csvFilename = ['Output_' datestr(now,'yyyy-mm-dd--HH-MM-SS') '.csv'];
csv.open(csvFilename);
header = {  'Step',
          'Disp Command (in)',
          'Displacement (in)',
          'Load (kips)',
          };
csv.writeHeader(header);

%% Process Commands
step = 0;   % Step counter
scr.writeFloat(F_SIM_BIT,1);
while (scr.readFloat(F_SIM_BIT) == 1)
   % Wait for OpenFresco to provide commands
   while (scr.readDouble(D_AVAILABLE) == 0)
   end
   % Pause if necessary
   if (scr.readFloat(F_PAUSE_BIT) == 1)
      input('Paused... hit any key to continue or CTRL-C to quit');
   end
   % Break loop if necessary
   if (scr.readFloat(F_SIM_BIT) == 0)
      break;
   end
   % Step counter increment
   step = step + 1;
   %disp(sprintf('Step %i',step));

   % Tell OpenFresco that the commands are being processed
   scr.writeDouble(D_STATE,PROCESSING);

   % Get the command from OpenFresco, scale and convert to DSP
   cmd = scr.readDouble(D_COMMAND);
   scaledcmd = cmd*SCALEFACTOR;   % should be in mm now
   scrcmd = scaledcmd / 500; % should be in DSP mm now
   %disp(sprintf('Command = %fin, scaled = %fin',cmd,scaledcmd));
   if ((scaledcmd >= UPPERLIMIT) || (scaledcmd <= LOWERLIMIT))
            disp(sprintf('COMMAND LIMIT!'));
```

```
        input('Paused... hit any key to continue or CTRL-C to quit');
    end

    % Write to SCRAMNet
    scr.writeFloat(F_COMMAND,scrcmd);
    pause(0.03);

    % Scale the load and displacement
    displacement = scr.readFloat(F_DISP_FEEDBACK)*500/SCALEFACTOR;
    %load = cmd*LOADSCALE;
    load = displacement*LOADSCALE;
    data = [step,
            cmd,
            displacement,
            load,
            ];
    csv.write(data);
    scr.writeDouble(D_DISP_FEEDBACK,cmd);
    scr.writeDouble(D_FORCE_FEEDBACK,load);
    scr.writeDouble(D_FEEDBACK_TIME,scr.readDouble(D_COMMAND_TIME));
    scr.writeDouble(D_STATE,DONE);
end
disp('EXIT');
scr.unmapScramnet;
clear scr;
```