Structural Engineering and Mechanics, Vol. 40, No. 1 (2011) 85-103
DOI: http://dx.doi.org/10.12989/sem.2011.40.1.085

Integrating OpenSees with other software - with application to coupling problems in civil engineering

Quan Gu[1a] and Ozgur Ozcelik*[2]

[1]*Department of Civil Engineering, School of Architecture and Civil Engineering, Xiamen University, Xiamen, Fujian, 361005, P.R. China*
[2]*Department of Civil Engineering, School of Engineering, Dokuz Eylul University, Buca/Izmir, 35160, Turkey*

(Received October 17, 2010, Revised May 23, 2011, Accepted June 24, 2011)

Abstract. Integration of finite element analysis (FEA) software into various software platforms is commonly used in coupling systems such as systems involving structural control, fluid-structure, wind-structure, soil-structure interactions and substructure method in which FEA is used for simulating the structural responses. Integrating an FEA program into various other software platforms in an efficient and simple way is crucial for the development and performance of the entire coupling system. The lack of simplicity of the existing integration methods makes this integration difficult and therefore entails the motivation of this study. In this paper, a novel practical technique, namely CS technique, is presented for integrating a general FEA software framework OpenSees into other software platforms, e.g., Matlab-Simulink® and a soil-structure interaction (SSI) system. The advantage of this integration technique is that it is efficient and relatively easy to implement. Instead of OpenSees, a cheap client handling TCL is integrated into the other software. The integration is achieved by extending the concept of internet based client-server concept, taking advantage of the parameterization framework of OpenSees, and using a command-driven scripting language called tool command language (TCL) on which the OpenSees' interface is based. There is no need for any programming inside OpenSees. The presented CS technique proves as an excellent solution for the coupling problems mentioned above (for both linear and nonlinear problems). Application examples are provided to validate the integration method and illustrate the various uses of the method in the civil engineering.

Keywords: coupling systems; OpenSees – Simulink; software integration; client-server techniques; soil-structure interaction; shake table modeling

1. Introduction

Finite Element Analysis (FEA) has been widely used in coupling systems as a powerful tool for predicting or simulating the behavior of structures subjected to static and/or dynamic loading conditions. Coupling systems usually include a FEA software and another software platform (e.g., Matlab, Optimization software, etc.), and can be used for problems in structural control, fluid-structure, wind-structure, soil-structure interactions and substructure method (Gawronski 1998, Cook

*Corresponding author, Assistant Professor, E-mail: ozgur.ozcelik@deu.edu.tr
[a]Associate Professor

et al. 2001, Paidoussis 2004). In these problems, the structural properties (e.g., geometric and material parameters) or the structural external load may depend on the structural responses which are the outputs of FEA (e.g., deformations or internal resisting forces of the structure). In an explicit method in which the state of the coupling system at a later time step is calculated from the state of the system at the current time step, the other software platform needs to call the FEA software at each time step, and incorporates the FEA results to determine and update the properties and/or loads of the FE model for performing the next-step analysis. Therefore, an efficient communication between FEA software and the other software platform is crucial for the performance of the entire coupling system.

In general, directly linking and compiling FEA software and another software platform together is extremely difficult, if not impossible. An intuitive way to achieve this coupling is to make the software platform the main program and enable it to run the FEA software whenever an FE analysis is needed. In this case, the FE model does not persist in the memory, and thus the model has to be reloaded/released before/after each call to perform FEA. After each call (usually at each time step), all state variables (e.g., the displacement, velocity, and acceleration of each node, stress, strain, and other stress states in each material) of the FE model need to be exported to a database or a file before the model is released from the computer's memory. Then at the beginning of the next call from the software platform, a restart process in FEA must be performed, and the model has to be reloaded into the memory with all the state variables imported from the database or the file. This method is obviously not efficient since it spends large amount of time in loading and releasing the FE model.

In the literature, alternative methods are used to achieve the coupling of FEA software and other software platforms. For example, in UI-SimCor hybrid simulation framework developed at the University of Illinois at Urbana-Champaign (Kwon *et al.* 2005), a simulation coordinator is employed to control and organize models associated with various platforms and/or actual experimental specimens, including an FEA software (e.g., ZeusNL, FEDEASLab, ANSYS, OpenSees, and Abacus), which are considered as super-elements with many DOFs (Elnashai *et al.* 2004). There are other alternative methods to integrate FEA software into various software platforms. One integration method is to compile a part or the entire FEA software into dynamic link library (DLL) or shared library as described in (Peng and Law 2002), so that the DLL may be integrated into other software platforms directly, or can be used as a web service program offering the FEA service for clients through internet. Another integration method is used in OpenFresco, in which the FEA software is coupled with the other software by making a special experiment element in FEA software as client and the other software platform as server and the communication is enabled by using a network socket (Yoshikazu and Fenves 2006, Schellenberg *et al.* 2006). In the internet online hybrid test system, a peer-to-peer (p2p) system is invented where a "coordinator" is required to achieve compatibility and equilibrium at boundaries among substructures (Pan *et al.* 2006). All these methods require either further programming work for FEA software (e.g., changing existing source code, or adding new experimental element, or recompiling FEA system, etc), or employing a heavy and complicated third party integration software platform to achieve the integration.

The integration could also be achieved by using a hand-shake mechanism to govern data exchange between FEA software and the other software as described in Cattarius (1999). This method takes advantage of the user subroutines in FEA software (e.g., Abaqus) and uses flag shared files for communication between the FEA software and the other software. This procedure involves

a large amount of alternate idle time between the two codes and thus is not efficient (Cattarius 1999). The desire of engineers and researchers for seeking a practical and simple method to integrate FEA software into other platforms entails the motivations of this study.

In this paper, a new integration method/technique is presented for integrating FEA software OpenSees into various platforms. OpenSees (Open System for Earthquake Engineering Simulation) is an open source FEA software used to model structural systems and simulates their earthquake responses (McKenna 1997, McKenna *et al.* 2004, http://opensees.berkeley.edu, http://archt.xmu.edu.cn/opensees/opensees.html). It is the main simulation platform for NEESGrid (http://www.nees.org) and also widely used in the academic world (McKenna *et al.* 2010). OpenSees' user interface is based on a command-driven scripting language called tool command language (TCL) which enables users to create versatile input files (Welch 2000). An advanced parameterization framework is recently implemented in OpenSees which allows users to visit or update the parameters of FE model easily by using TCL commands (Scott and Haukass 2008). The new integration method presented herein does not require any programming work within OpenSees's source code. Instead users are required to write simple TCL scripts to achieve the communication between OpenSees and another software platform which will be explained in detail later.

In order to describe this integration method, the concept of the internet based client-server model is borrowed and extended. The client, instead of internet users, is a cheap/tiny piece of code plugged into another software platform. OpenSees acts like a server program by making the finite element (FE) model persistent in the memory, receiving, and executing commands from the client. The client is usually a component of other software platforms such as a C++ object or a global variable which can be created by using the API of the other software platforms. By using TCL scripts, the client creates and holds a connection with OpenSees and thus the communication between OpenSees and the other software platform is established. A simple way to achieve this connection is by using a simple TCL network communication channel (or socket) based on TCP or other network protocols (e.g., UDP). The new integration method/technique is named CS (client-server) technique for naming convenience, however it is significantly different from the classical concept of the Internet based client-server model in the sense that it is used for integration purpose and thus is limited to one-client-one-server mode and usually both client and server are on the same computer.

2. Integration of OpenSees into other software platforms by using CS technique

In this section, coupling OpenSees with Simulink is presented as an illustration example showing the use of the CS technique. However, OpenSees may be integrated into other software platforms using the same technique as described herein. In general cases, the TCL-based cheap client needs to be integrated into the software platform using a similar method described here. Simulink is a platform for multi-domain simulation and model-based design for dynamic systems. The interactive graphical environment and a customizable set of block libraries in Simulink enable it to be extended for various applications. For this application example, the client is a persistent C++ object created by and stored in a user-defined S-function in Simulink.

The integration of OpenSees with Simulink is described in Fig. 1. Both OpenSees and Simulink are running in parallel on the same computer. Following steps are performed (refer to Appendix A for programming details):

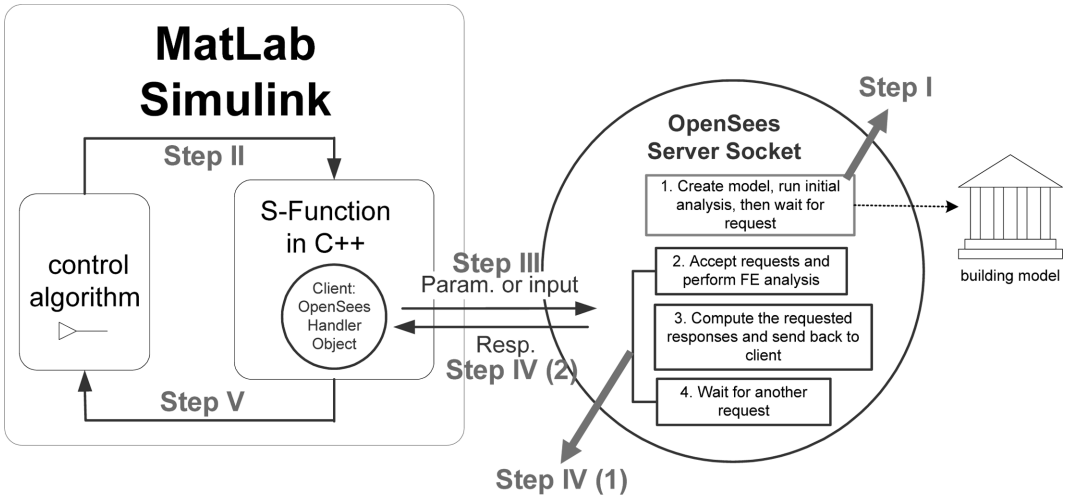

Fig. 1 Simulink – OpenSees integration by using CS technique

Step I: At the very beginning, on the server side, OpenSees creates a finite element model and performs necessary initial analysis (e.g., gravity load analysis). Then, it stops and waits for commands from Simulink for further actions.

Step II: After OpenSees is set up as shown in Step I, Simulink sets up the connection to OpenSees server through its plugged-in client and holds the connection in its S-function. Currently, this connection is achieved by using a network socket. Simulink then begins to solve a mathematical model of a physical problem (e.g., shake table model) programmed within Simulink by block diagrams.

Step III: When Simulink needs to perform FEA, it sends an analysis request to OpenSees together with a set of FE model parameters and/or input loads (e.g., earthquake acceleration) by calling the S-function. Note that within S-function, the client forms and sends the associated command to OpenSees using the established connection. It will then wait for OpenSees to run the requested analysis and send back the structural response needed to advance the entire model.

Step IV: On the server side, once OpenSees receives commands from the client for a specific action with new model parameters and/or loads, it performs the following actions: (1) updates the model by using the parameterization framework of OpenSees, performs the requested analysis, gets the required structural responses (e.g., resisting force), and (2) sends them back to the client through the same connection (and thus to Simulink). These actions are achieved by using TCL commands and taking advantage of the existing parameterization framework in OpenSees. After these actions are completed, OpenSees stops and waits for further commands from the client.

Step V: After Simulink receives these responses, it incorporates these FEA results into the Simulink simulation, and continues to advance the mathematical model. When Simulink needs to perform FEA with a new set of parameters and/or input loads, the steps (III) to (V) are repeated.

The implementation details of above outlined procedure are given in Appendix A.

3. Advantages and disadvantages of CS technique

Advantages of integrating OpenSees into various software platforms by using the CS integration are summarized below:

(i) It is very easy to program and maintain. Both software platforms keep their main flowchart unchanged. On the OpenSees server side, a very limited amount of programming work using TCL is required and there is no need to interfere with the OpenSees' source code. On the client side, instead of integrating OpenSees software, a cheap/tiny client is integrated into the other software platform by using their application programming interface (API), therefore not increasing the complexity of the entire coupling system. The maintenance (e.g., debugging, further development, or improvement) of the TCL code and the client side code is usually very easy. Compared with other general integration framework like UI-SimCor hybrid simulation framework, the presented CS technique is much easier for users to implement and maintain considering that UI-SimCor needs a heavy third-party software to deal with various platforms (Kwon *et al*. 2005).

(ii) It is efficient. The OpenSees model is persistent in the memory and thus there is no need to release or reload the model during the entire analysis process. The connection between client and OpenSees server is established through a fast speed socket channel. Furthermore, a small amount of data needs to be transferred between client and server. Only necessary data requested by the software platform (e.g., specified nodal displacements which are a set of real numbers) and those sent back to FEA software (e.g., an updated acceleration at one time step which is a real number) are transferred. The demonstration examples below show that the extra time due to the communication and transferring data between OpenSees and other platforms is almost negligible. Further study of which the details are not documented here shows that for more complex coupling systems where moderate amount of data (e.g., 1~5 Mb) is transferred between the client and server, extra time needed for communication and data transfer between the two platforms is much less than the computational time and thus is negligible.

(iii) It is flexible. Users are allowed to update the properties of the FE model or loading parameters at any time step by using the parameterization framework in OpenSees. And the cheap/tiny client can be flexibly integrated into various software platforms by using their API. Furthermore, although not necessary for the CS technique to work, by enhancing the OpenSees software with new commands such as "*tryOneStep*" (i.e., getting converged solution but not updating the variables of previous step by those of the current step), "*revertToLastStep*", and "*commit*" (i.e., updating the variables of previous step by those of current step), the CS technique is able to work together with implicit algorithms (e.g., Newton iterative algorithm) inside the other software platform.

(iv) It is robust. Connection between the client and server is set up only once by the network socket using TCP/IP or other protocol (e.g., UDP) at the beginning of the analysis and is held at all time (refer to step II in section 2). Once the connection is set up, it provides a reliable communication method due to the robustness of the internet protocol (note that both the client and the server are on the same computer).

(v) It has a broad application area. OpenSees is a powerful FEA software framework and widely used in the academic and engineering disciplines. The CS technique allows OpenSees to be integrated in a wide range of coupling problems where FEA software is required to simulate structural responses.

When compared with OpenFresco (Schellenberg *et al*. 2006) in which FEA is a client instead of a server, the presented integration method has different application areas: when OpenSees analysis engine is used as a server, it allows the client (thus the other software platform) to be the main program, controlling the flowchart of the entire coupling system. On the contrary, if OpenSees is set as a client (as in OpenFresco), it will always be the main program, with its rigidly structured FEA software framework being the main flowchart of the entire system. This may limit the range of application of the coupling system; in other words, it may not be able to accommodate the case that the other software's flowchart is the main program flowchart of the entire coupling system.

Disadvantages of integrating OpenSees into various software platforms by using the CS integration are summarized below:

(i) Users need to learn basic programming skills using TCL. The cheap/tiny client on the other software platform needs to handle TCL commands, thus usually the client needs to be linked and compiled with the TCL library.

(ii) The cheap/tiny client needs to be integrated into other software platforms by using their API. This could be a limitation if the other software does not have suitable API for this integration.

(iii) Although both OpenSees and the other software are in the same machine, an internet protocol (e.g., TCP/IP) is required due to the fact that the communication between them is achieved by using a network socket.

(iv) The data transferred between OpenSees and the other software is a string stream in ASCII format and usually very small (much less than 1 Kb). However for some very special cases in which the amount of transferred data is large (e.g., the client asked for the entire stiffness matrix of structure, which may be larger than 10 MB), the current communication method needs to be improved. One solution to overcome this difficulty is to develop or use new protocols and using formatted binary data for the data exchange between OpenSees and other platform (e.g., NHCP protocol) (Cowart *et al*. 2007).

It is worth mentioning that the presented CS integration technique should be understood as a modification/extension of the concept of the Internet-enabled distributed services framework based on web services model developed (Peng and Law 2004). However in the presented integration method, OpenSees provides services to another software instead of computer users. More importantly, it takes advantage of the powerful existing TCL functions and the parameterization framework of OpenSees to change OpenSees into a service program without interfering with its source code. Furthermore, the CS technique is focused on the integration, and thus uses one-server-one-client in local communication mode (i.e., both OpenSees and the other platforms are on the same machine, and thus poses no risk of communication loss due to possible internet failure).

It should be noted that the presented integration method can be potentially extended to intranet or internet with the same programming method except that in the "*socket*" command, the remote machine's IP address needs to be provided (refer to Appendix A). This technique as well as other techniques (e.g., OpenFresco) can potentially be used in hybrid physical testing, where part of the model is physically tested, and part is numerically modeled. In this case, the OpenSees will be the server and the client will be the physical model. However, the communication between OpenSees and the other software platform may be disconnected due to the fact that the internet connection may be lost. Thus the CS technique used in distributed systems needs further development. As mentioned earlier, one possible solution is to develop special protocols to handle the case of internet failure, such as the one used in NHCP (Cowart *et al*. 2007).

4. Application examples

In this section, three application examples are presented to illustrate the various applications of the CS technique.

4.1 Example 1. Linear shake table - linear structure interaction

The first example is used to verify the coupling system, by comparing the results obtained based on CS technique with those obtained analytically (i.e., closed form solution). A shake table with a specimen mounted on its rigid platen is modeled by a coupling system composed of Simulink and OpenSees where a simple model of a shake table (servo-hydraulic system) is programmed in Simulink and a finite element model of a linear single-degree-of-freedom (SDOF) specimen is built using OpenSees. The integration of OpenSees and Simulink is achieved by using the CS technique. The transfer function estimated using the coupling system is compared with its counterpart obtained analytically using Simulink-only in which both shake table and specimen are modeled (note that in this example, the linear elastic specimen model is simple enough to model in Simulink). Considering that the main focus of this paper is about the integration of two software platforms, the details regarding the modeling of the shake table and a SDOF specimen are not fully presented here; however interested readers can refer to the related literatures (Ozcelik 2008, Williams *et al.* 2001, Shortreed *et al.* 2001, Crewe and Severn 2001, Conte and Trombetti 2000, Thoen and Laplace 2004).

The block diagram for a linear servo-hydraulic shake table system with a proportional-integral-derivative (PID) controller and a force feedback stabilizing term is given in Fig. 2 (Ozcelik 2008). The closed-form formulation of the closed-loop table transfer function between the actuator force and the reference displacement signal can be obtained from Fig. 2 by algebraic manipulations. In Fig. 2, $G_h(s)$ is the transfer function between flow into the actuator and actuator force, where s is the Laplace variable, and $G_{xf}(s)$ is the transfer function between actuator force f and table displacement x, which includes the dynamics of the SDOF specimen mounted on the rigid platen

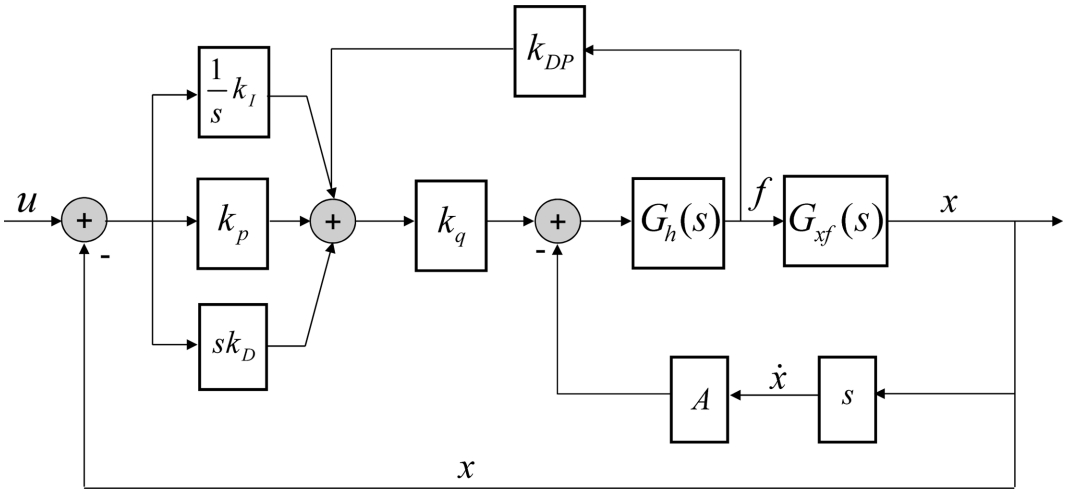

Fig. 2 A linear table with controller - linear specimen interaction model in Simulink (Simulink only model)

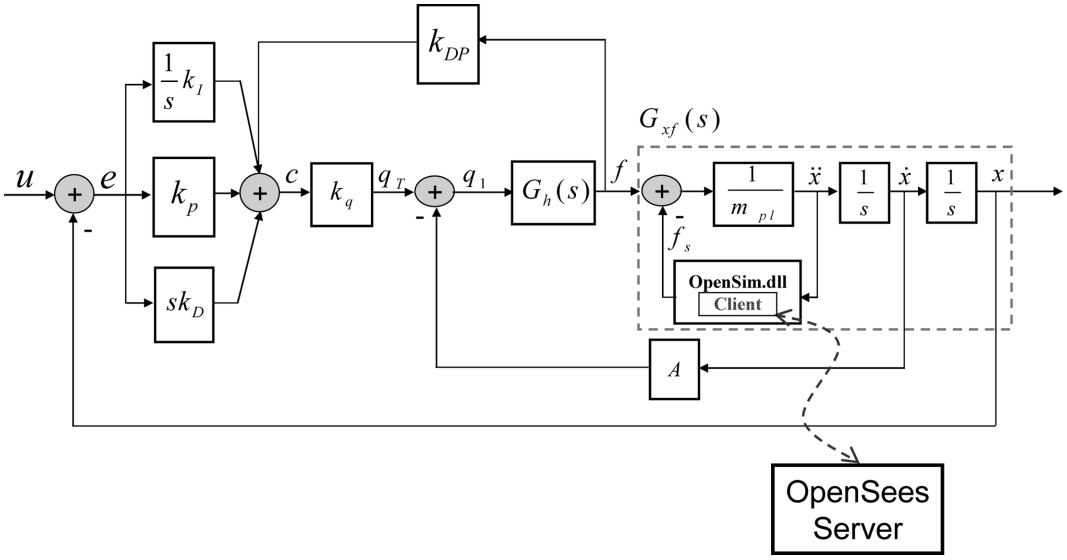

Fig. 3 A linear table with controller - linear specimen interaction model in OpenSees -Simulink coupling system achieved by using the CS technique

(thus including the interaction between specimen and shake table). The remaining model parameters shown in Fig. 2 are: A = effective actuator piston area, k_q = linearized flow gain, k_P = proportional, k_D = derivative, k_I = integral, and k_{DP} = force-feedback control gains (Ozcelik 2008).

Magnitude-phase responses of the transfer function of the closed-loop system shown in Fig. 2 can be obtained analytically (i.e., in closed-form). The following model parameters are used in this example $A = 0.3324$ m^2, $k_q = 3600$ lit/min/Volts, m_{pl} = 144 ton, $k_P = 1.50$ V/V, k_{DP} = –0.15 V/V, $k_D = 0.0$ V/V and k_I = 0.0 V/V. The dynamic parameters of the SDOF system are m_s = 65 ton, ξ_s = 3%, ω_s = 12.5664 rad/s.

Using the CS technique, SDOF specimen can be modeled with OpenSees instead of Simulink so that the powerful analysis and modeling capabilities of OpenSees can be taken advantage of, which greatly improves the modeling capability of the coupling system (i.e., in order to potentially model the complicated nonlinear structural system). The block diagram of the coupling system (a linear shake table modeled with Simulink and SDOF linear specimen modeled with OpenSees) is shown in Fig. 3. Inside the block of $G_{xf}(s)$ in Fig. 3 (i.e., block in dashed-line), a client is created by OpenSim.dll, and then used in the coupling system to replace the part of specimen model in Simulink. Once the platen acceleration $\ddot{x}$ is known, inertial forces can be calculated and the specimen resisting force f_s can be computed by OpenSees. By using OpenSees' capacity of modeling nonlinear systems, real complicated RC buildings on shake table can be modeled using the method presented herein.

Fig. 4(a) and 4(b) show the magnitude and phase response plots, respectively, of the model obtained by running the simulation under a band-limited white noise (WN) input (i.e., [0.25-20] Hz), with root-mean-square (RMS) amplitude equal to 0.13 g. The actuator force f obtained by analyzing the Simulink-OpenSees model (i.e., coupling system) was combined with the input u to estimate a transfer function between u and f. The magnitude and phase plots of the transfer function as a function of input frequency are compared with those obtained analytically.

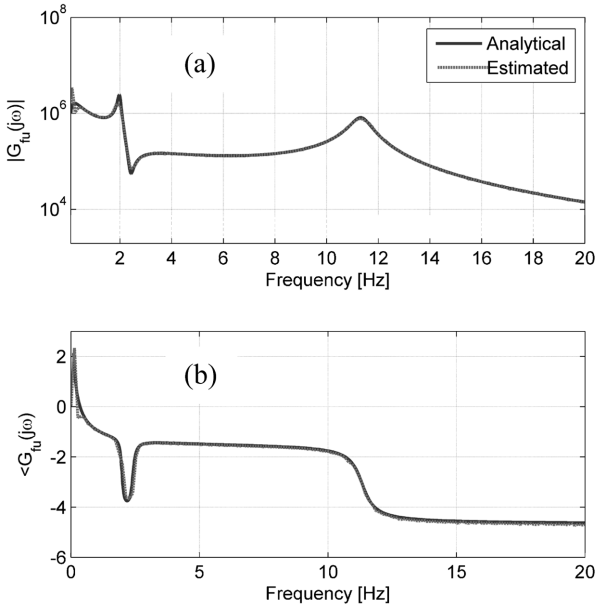

Fig. 4 Comparisons of the (a) magnitude and (b) phase responses of the analytical and numerical transfer functions of the linear shake table model with a linear specimen

As shown in Fig. 4, the analytically and numerically obtained transfer functions match almost perfectly, implying that the CS technique used in the coupling system is trustable. Discrepancies observed at very low frequencies are due to estimation errors. WN acceleration input is band-limited, and therefore the system is not excited at very low frequency range (i.e., <0.25 Hz). Also, the peak and notch pair observed around the natural vibration frequency of the SDOF specimen at 2 Hz is slightly missed in the estimated results, which can be attributed to the resolution problem in the estimation procedure due to the length of the recorded shake table response in time domain.

In order to study the efficiency of the communication between OpenSees and Simulink, the accelerations imposed on the OpenSees model as well as the responses (i.e., resisting force of the structure) of the structure are recorded while solving the coupling system. These recorded platen accelerations are then imposed on the same specimen model in OpenSees-only model (i.e., no Simulink model of the shake table), the resisting forces from the OpenSees-only model are compared with the recorded resisting forces from the coupling system. It is observed that the two forces are in perfect agreement. The computational time of the OpenSees-only analysis is about 90% of that used in the coupling system. It should be noted here that, the Simulink model of the shake table is linear and very simple, and thus its computational time is almost negligible when compared with the total computational time of the coupling system. This shows that communication between the Simulink and OpenSees using CS technique is very efficient.

4.2 Example 2. Nonlinear shake table - nonlinear structure interaction

The complex dynamics of large shake table systems emanate from multiple dynamic interactions and nonlinearities among various system components (Conte and Trombetti 2000, Thoen and Laplace 2004, Ozcelik 2008, Dyke *et al.* 1995, Trombetti and Conte 2002, Ozcelik *et al.* 2008,

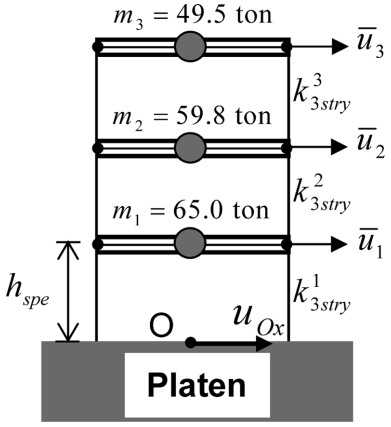

Fig. 5 Three-story shear frame mounted on the shake table platen

Table 1 Physical characteristics of the three story shear-frame

	Mass [kg]	k^i_{3stry} [MN/m]	Story height [m]
1st Story	65,000	76.3636	2.75
2nd Story	59,760	61.0909	2.75
3rd Story	49,540	61.0909	2.75
Total	174,300	N/A	8.25

Table 2 Modal analysis results for the linear elastic undamped one-bay 3 story shear-frame

Mode #	Natural frequency [Hz]	Natural period [sec]	Effective modal mass ratio [%]
1	2.51	0.40	89.00
2	6.67	0.15	9.57
3	9.26	0.11	1.43

Ozcelik *et al.* 2008, Luco *et al.* 2010). The platforms of this example are the same as those in the Example 1 (i.e., Simulink and OpenSees) except that both shake table and the specimen are modeled as complicated nonlinear systems. The details of the nonlinear shake table model with a controller composed of feedback as well as feedforward terms can be found in the literature (Ozcelik 2008, Ozcelik *et al.* 2008, Ozcelik *et al.* 2006).

In order to investigate the nonlinear table - nonlinear specimen interaction problem, a two-dimensional three story shear frame is used as the specimen (Fig. 5) and modeled using OpenSees. Beams are considered rigid to enforce a typical shear-building behavior and therefore only one horizontal degree-of-freedom (DOF) is assigned to each floor. Floor masses are assumed to be lumped at floor levels. Physical characteristics of the shear-frames are summarized in Table 1. The parameters k^i_{3stry} is the initial story stiffness of each floor. Natural frequencies, natural periods, and effective modal mass ratios for the undamped structure are given in Table 2. Viscous damping in the form of Rayleigh damping is assumed with damping ratio ξ = 0.03 for the first and second modes of vibration.

The story shear force – interstory drift relation for the columns are modeled using the nonlinear Menegotto-Pinto (M-P) material constitutive model (Filippou *et al*. 1983, Barbato and Conte 2006). The model parameters are: E = 2.1E5 MPa, b = 0.10; R_0 = 200, a_1 = 18.5, a_2 = 0.15, $a_3 = a_4$ = 0.0. Elastic story shear forces obtained using the Downtown Los Angeles area are reduced by a force reduction factor of $R4$ = 4.0. Thus, the reduced story initial yield strengths for the shear frame become: 510.0 kN, 400.6 kN, 211.1 kN for the first, second and third story, respectively.

The total base shear force acting on the shake table platen at point O (Fig. 5) can be calculated using the absolute specimen acceleration response retrieved from OpenSees at each time step. The equation of motion of the platen with respect to point O (Fig. 5) can be written as (the specimen's dynamic effects are also included)

$$\ddot{u}_O(t) = M_O^{-1}\{f(t) - f_s(t)\} \tag{1}$$

where M_O is the effective mass of the shake table platen (assumed rigid), nonlinear function $f(t)$ is the total actuator force acting on the platen, nonlinear function $f_s(t)$ is the total specimen shear force, which is calculated using OpenSees. The Simulink implementation of the nonlinear shake table with a nonlinear MDOF specimen has the same implementation topology as that of the linear case, however their implementation details are different since the nonlinear relations and a different control law are used for modeling the shake table and the specimen system. In this example, the partial differentiation equation (PDE) solver used in Simulink is the 4th order Runge-Kutta Method (Ozcelik 2008), which requires function evaluations (call to OpenSees) at intermediate integration time steps. The seamless integration of OpenSees into Matlab Simulink by using CS technique achieves this complicated coupling system.

Fig. 6 shows the estimated magnitude response of the transfer function between the actuator force as input and the platen acceleration as output. Transfer functions with solid and dashed lines are the coupled models with a linear elastic specimen (R1) and a nonlinear specimen (R4), respectively (Fig. 6). From Fig. 6, it is observed that three different peak and notch pairs occur at the vicinity of the natural frequencies of the linear specimen (R1). This example illustrates that the newly developed CS technique can be used to couple complicated nonlinear systems modeled using two different simulation platforms.

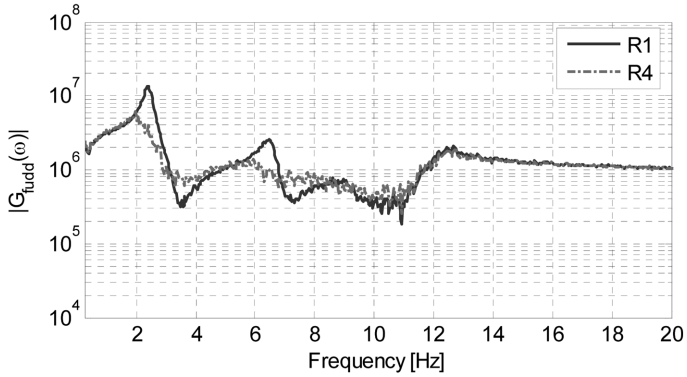

Fig. 6 Comparison of the magnitude plots of the estimated $G_{fudd}(s)$ for the nonlinear shake table model with a linear specimen (R1) and a nonlinear specimen (R4)

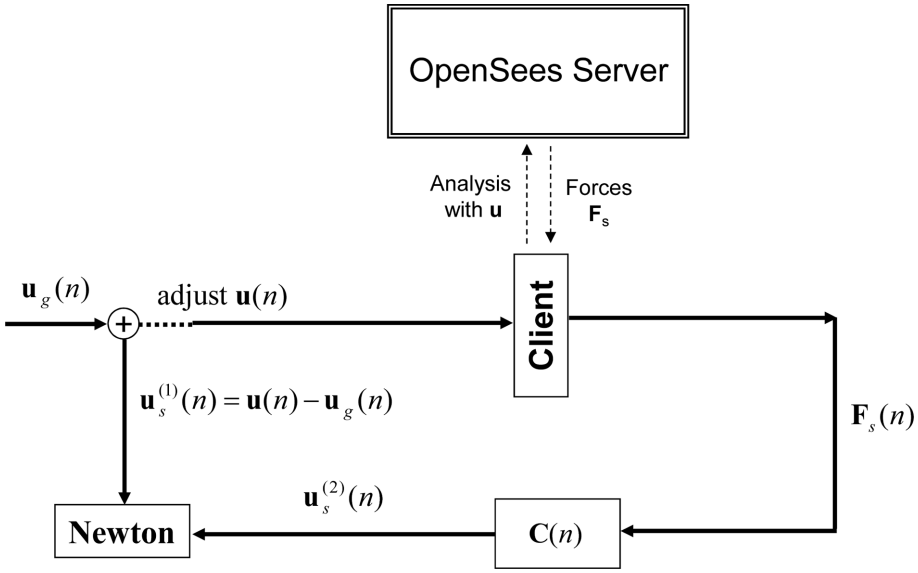

Fig. 7 A SSI analysis framework based on the substructure method in time domain, (achieved by using the CS technique)

4.3 Example 3. Soil structure interaction (SSI) problem

In this application, a new analysis framework is presented which solves SSI problems based on the substructure method in time domain. In this framework, the analytically obtained frequency dependent compliance functions of a rigid foundation sitting on a uniform or layered linear elastic half-space is modeled in time domain using discrete-time recursive filters (Safak 2006) and the linear or nonlinear superstructure is modeled by the FEA software OpenSees. The substructure method in block diagram format for soil-structure interaction problems is shown in Fig. 7 (Luco 1980). The total response **u** of the rigid foundation at the center of the foundation can be written as

$$\mathbf{u} = \mathbf{u}_s + \mathbf{u}_g \tag{2}$$

where $\mathbf{u}_g$ is the recorded free field input motion and $\mathbf{u}_s$ is the additional motion of the foundation imposed by the generalized forces and moments that the rigid foundation exerts on the soil. The foundation displacement **u** excites the foundation and thus the structural system, and the force $\mathbf{F}_s$ from the structure to the soil is obtained by OpenSees analysis as a function of **u**. The motion $\mathbf{u}_s$ can be calculated in the frequency domain by

$$\mathbf{u}_s = \mathbf{C}(\omega)\mathbf{F}_s(\mathbf{u}) \tag{3}$$

where $\mathbf{C}(\omega)$ is the compliance matrix of the rigid foundation embedded in a soil medium (Luco 1980, Mita and Luco 1989, Luco and Apsel 1983, Luco and Mita 1987, Sieffert and Cevaer 1991).

The frequency dependent impedance functions (i.e., inverse of the compliance matrix) can be represented in time domain by discrete time recursive filters as described in Safak (2006). In this method, impedance functions of various soil-foundation assemblies are represented with discrete-time recursive filters using an estimation procedure in the frequency domain. By taking the inverse Z-transform of these filters, the frequency dependent impedance functions (or compliance functions,

$\mathbf{C}(\omega)$) can be transformed into time domain as difference equations (difference equations are linear combinations of the current and past time steps soil responses), which in turn can be used in standard time history analysis (Safak 2006).

The unknown foundation displacement $\mathbf{u}$ and $\mathbf{u}_s$ in Eqs. (2) and (3) can be solved iteratively using an implicit Newton algorithm (Hildebrand 1974), as shown in Fig. 7. For each iteration, given a trial input displacement $\mathbf{u}$, OpenSees performs one-step base excitation analysis and obtains $\mathbf{F}_s$, the force exerted to the soil by the structure. Then $\mathbf{u}_s^{(2)} = \mathbf{C}(\omega)\mathbf{F}_s(\mathbf{u})$ in Eq. (3) can be calculated by the soil filters represented in time domain. Newton algorithm computes the residuals between $\mathbf{u}_s^{(1)}$ and $\mathbf{u}_s^{(2)}$, and gets a new trial displacement $\mathbf{u}$ based on its updating rule (Hildebrand 1974). This iteration process continues until the convergence is reached. As shown in Fig. 7, a cheap client is integrated into the presented SSI framework for achieving efficient communication between SSI system and OpenSees at each time step.

As shown in Fig. 8, a SSI system with one-bay one-story frame structure on a surface rigid square foundation sitting on a uniform elastic half space is taken as an application example. The frame is modeled with linear elastic beam/column elements with $H = 6.3$ m, $L = 25.2$ m, and the equivalent column section width $b = 6.24$ m. The lumped mass at floor level is 7,550 tons. Elastic modulus for columns and beam are $E_c = 2.7 \times 10^{10}$ N/m^2, $E_b = 2 \times 10^{14}$ N/m^2, respectively. The first and second natural vibration frequencies of the structure are $f_1 = 2.33$ Hz and $f_2 = 13.94$ Hz, respectively. Rayleigh damping is used with viscous damping ratio ξ = 1% for the 1st and 2nd modes. For the soil, the mass density is $\rho = 1.874 \times 10^3$ kg/m^3, the shear wave velocity is $\beta = 400$ m/s, and the hysteretic damping ratios are $\xi_\beta = 0.001$ and $\xi_\alpha = 0.0005$, Poisson's ratio is $\upsilon = 1/3$.

Fig. 9 shows the comparison between the analytical and estimated compliance functions where C_{HH}, C_{MM}, and C_{HM}(=C_{MH}) are horizontal, rocking and coupling compliance functions, respectively. An implicit method based on Newton's method is used in the analysis with integration time step of Δt = 0.0039 seconds (i.e., 1/256 seconds).

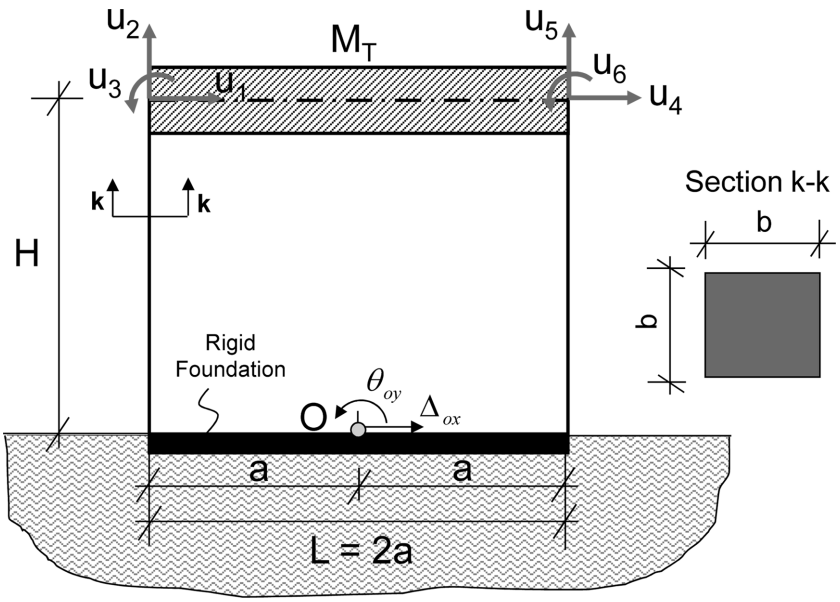

Fig. 8 Frame structure sitting on the rigid surface foundation embedded in half-space soil medium

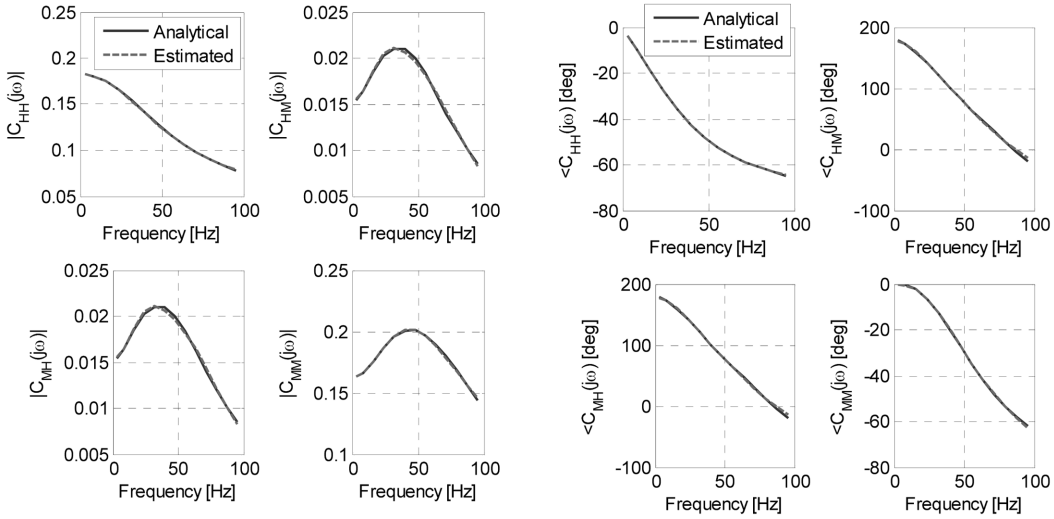

Fig. 9 Comparison of the magnitude and phase responses of the analytically and numerically estimated frequency dependent compliance functions

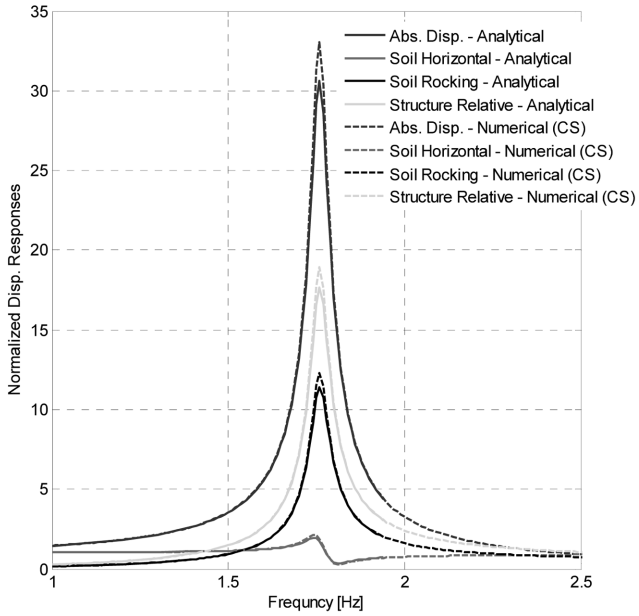

Fig. 10 Comparison of normalized analytical vs. numerical displacement frequency response functions of various response quantities

In Fig. 10, the analytical solutions for the horizontal responses of the foundation-soil system, normalized by the amplitude of the free-field ground motion, are compared with their numerical counterparts using the SSI time domain solution framework. The analytical solutions can be found in literature (Luco 2005).

The resonance frequency obtained by the two methods agrees perfectly (i.e., 1.76 Hz). Amplitude errors for the absolute and relative displacement of the structure, and the translational and the rocking motions of the soil are 8%, 7%, 8%, and 1%, respectively. These differences might be attributed to the fact that structural model is not an ideal SDOF system, Rayleigh damping can specify damping ratios for only two different natural frequencies, and there are discrepancies between the estimated and real compliances due to the accuracy of the estimation procedure. However, the results are acceptable for practical purposes.

5. Conclusions

Integration of finite element analysis (FEA) software into other software platforms is commonly used in coupling systems such as structural control, fluid-structure, wind-structure, soil-structure interactions and substructure method where FE analysis must be used for obtaining structural responses. In this study, a new integration technique, named CS technique, is presented for integrating a general purpose FEA software OpenSees into various other software platforms. By extending the internet based client-server concept, taking advantage of the powerful TCL language on which the OpenSees' interface is built, and using the advanced parameterization framework of OpenSees, the CS technique greatly simplifies the integration problem. With this technique, only a cheap/tiny client is required to be integrated into the other software platform. There is no need for any programming work in OpenSees. The integration method presented is efficient, flexible, robust and very easy to implement, therefore provides an ideal solution for engineers and researchers seeking a practical and simple method to integrate OpenSees into other platforms. The limitation of this method is that the cheap client may not be easily integrated into other software if the software does not have suitable API for this integration. Implementation and uses of the CS technique are illustrated and verified using three application examples, involving integration of OpenSees into Matlab Simulink® and a novel soil structure interaction (SSI) system for solving both linear and nonlinear coupling/interaction problems. The presented CS technique proves as an excellent solution for these types of coupling problems.

Acknowledgements

The research was partially supported by the Fundamental Research Funds for the Central Universities of China under Award No. 2010111075. This support is gratefully acknowledged. The writers would like to thank Professor Joel P. Conte and Professor J. Enrique Luco from University of California at San Diego for their invaluable discussions and advices during this research work.

References

Barbato, M. and Conte, J.P. (2006), "Finite element structural response sensitivity and reliability analyses using smooth versus non-smooth material constitutive models", *Int. J. Reliab. Saf.,* **1**, 3-39.

Cattarius, J. (1999), "Numerical wing/store interaction analysis of a parametric F16 wing", Ph.D. Dissertation, Virginia Polytechnic Institute and State University.

Cook, R.D., Malkus, D.S., Plesha, M.E. and Witt, R.J. (2001), *Concepts and Applications of Finite Element Analysis*, 4th Edition, John Wiley & Sons Inc., NY.
Conte, J.P. and Trombetti, T.L. (2000), "Linear dynamic modeling of a uni-axial servo-hydraulic shaking table system", *Earthq. Eng. Struct. D.,* **29**(9), 1375-1404.
Cowart, C., Hubbard, P., Miller, L. and Crawford, G. (2007), *NHCP Reference Implementation and Protocol Reference Guide,* Version 1.0, NEES Cyber-Infrastructure Center, University of California, San Diego.
Crewe, A.J. and Severn, R.T. (2001), "The European collaborative programme on evaluating the performance of shaking tables", *Phil. Trans. R. Soc. Lond A,* **359**, 1671-1696.
Dabney, J. and Harman, T.L. (2004), *Mastering Simulink*, Pearson/Prentice Hall, NJ.
Dyke, S.J., Spencer, B.J., Quast, P. and Sain, M.K. (1995), "Role of control-structure interaction in protective system design", *J. Eng. Mech.*, **121**(2), 322-338.
Elnashai, A., Spencer, B., Kuchma, D., Ghaboussi, J., Hashash, Y. and Gan, Q. (2004), "Multi-axial full-scale sub-structured testing and simulation (must-sim) facility at the University of Illinois at Urbana-Champaign", *Proceeding of the 13th World Conference on Earthquake Engineering,* Vancouver, Canada, August.
Filippou, F.C., Popov, E.P. and Bertero, V.V. (1983), *Effects of Bond Deterioration on Hysteretic Behavior of Reinforced Concrete Joints*, Report EERC 83-19, Earthquake Engineering Research Center, University of California, Berkeley.
Gawronski, W.K. (1998), *Advanced Structural Dynamics and Active Control of Structures*, Springer-Verlag, NY.
Hildebrand, F.B. (1974), *Introduction to Numerical Analysis*, 2nd Edition, McGraw-Hill, NY.
Kwon, O.S., Nakata, N., Elnashai, A.S. and Spencer, B.A. (2005), "Framework for multi-site distributed simulation and application to complex structural systems", *J. Earthq. Eng.*, **9**(5), 741-753.
Luco, J.E. (1980), *Seismic Safety Margins Research Program, Linear Soil-structure Interaction*, Lawrence Livermore Laboratory, California, UCRL-15272.
Luco, J.E. and Apsel, R.J. (1983), "On the green's functions for a layered half-space: Part I", *Bull. Seismol. Soc. Am.*, **73**, 909-929.
Luco, J.E. and Mita, A. (1987), "Responses of the circular foundation on a uniform half-space to elastic waves", *Earthq. Eng. Struct. Dyn.*, **15**, 105-118.
Luco, J.E. (2005), *Soil-Structure Interaction: Class Notes*, University of California San Diego.
Luco, J.E., Ozcelik, O. and Conte, J.P. (2010), "Acceleration tracking performance of the NEES-UCSD shake table", *J. Struct. Eng.*, **136**(5), 481-490.
McKenna, F. (1997), "Object-oriented finite element programming: frameworks for analysis, algorithms and parallel computing", Ph.D. Dissertation, University of California, Berkeley.
McKenna, F., Scott, M.H. and Takahashi, Y. (2004), "An object-oriented software environment for collaborative network simulation", *Proceeding of the 13th World Conference on Earthquake Engineering*, Vancouver, Canada, August.
McKenna, F., Scott, M.H. and Fenves, G.L. (2010), "Nonlinear finite element analysis software architecture using object composition", *J. Comput. Civil Eng.*, **24**(1), 95-107.
Mita, A. and Luco, J.E. (1989), "Impedance functions and input motions for embedded square foundations", *J Geotech. Eng.*, **115**(4), 491-503.
Ozcelik, O. (2008), "A mechanics-based virtual model of NEES-UCSD shake table: theoretical development and experimental validation", Ph.D. Dissertation, University of California, San Diego.
Ozcelik, O., Luco, J.E., Conte, J.P., Trombetti, T.L. and Restrepo, J.I. (2008), "Experimental characterization, modeling and identification of the UCSD-NEES shake table mechanical system", *Earthq. Eng. Struct. D.*, **37**, 243-264.
Ozcelik, O., Luco, J.E. and Conte, J.P. (2008), "Identification of the mechanical subsystem of the NEES-UCSD shake table by a least-square approach", *J. Eng. Mech.*, **134**(1), 23-34.
Ozcelik, O., Conte, J.P. and Luco, E.J. (2006), "Virtual model of the NEES-UCSD high performance outdoor shake table", *Proceeding of the 4th World Conference on the Structural Control and Monitoring*, San Diego, July.
Paidoussis, M.P. (2004), *Fluid-structure Interactions, Slender Structures and Axial Flow*, Vol. 2, Elsevier Academic Press.
Pan, P., Tomofuji, H., Wang, T., Nakashima, M., Ohsaki, M. and Mosalam, K.M. (2006), "Development of peer-

to-peer (p2p) internet online hybrid test system", *Earthq. Eng. Struct. D.*, **35**, 867-890.
Panagiotou, M.P. and Restrepo, J. (2007), *Computational Model for the UCSD 7-story Structural Wall Building Slice*, Report SSRP-07/09, University of California, San Diego.
Peng, J. and Law, K.H. (2002), "A prototype software framework for internet-enabled collaborative development of a structural analysis program", *Eng. Comput.*, **18**, 38-49.
Peng, J. and Law, K.H. (2004), "Building finite element analysis programs in distributed services environment", *Comput. Struct.*, **82**, 1813-1833.
Safak, E. (2006), "Time-domain representation of frequency-dependent foundation impedance functions", *Soil Dyn. Earthq. Eng.*, **26**, 65-70.
Schellenberg, A., Mahin, S. and Fenves, G.L. (2006), "Software framework for hybrid simulation of large structural systems", *Proceedings Structures Congress, ASCE*, Long Beach.
Scott, M.H. and Haukaas, T. (2008), "Software framework for parameter updating and finite element response sensitivity analysis", *J. Comput. Civil Eng.*, **22**(5), 281-291.
Shortreed, J.S., Seible, F., Filiatrault, A. and Benzoni, G. (2001), "Characterization and testing of the caltrans seismic response modification device test system", *Phil. Trans. R. Soc. Lond. A*, **359**, 1829-1850.
Sieffert, J.G. and Cevaer, F. (1991), *Handbook of Impedance Functions*, France Quest Editions, Presses Academiques, Paris.
Thoen, B.K. and Laplace, P.N. (2004), "Offline tuning of shaking table", *Proceeding of the 13th World Conference on Earthquake Engineering*, Vancouver, Canada, August.
Thoen, B.K. (2004), *469D Seismic Digital Control Software*, MTS Corporation, MN.
Trombetti, T.L. and Conte, J.P. (2002), "Shaking table dynamics: results from a test analysis comparison study", *J. Earthq. Eng.,* **6**(4), 513-551.
Welch, B.B. (2000), *Practical Programming in Tcl and Tk*, 3rd Edition, Prentice-Hall Inc., NJ.
Williams, D.M., Williams, M.S. and Blakeborough, A. (2001), "Numerical modeling of a servohydraulic testing system for structures", *J. Eng. Mech.*, **127**(8), 816-827.
Yoshikazu, T. and Fenves, G.L. (2006), "Software framework for distributed experimental-computational simulation of structural systems", *Earthq. Eng. Struct. Dyn.*, **35**, 267-291.

Appendix A

In this section, the program implementation of the presented CS technique is illustrated by an example, coupling OpenSees with Matlab-Simulink® (denoted as Simulink herein).

Server side: At the very beginning of the analysis of the entire coupling system (refer to step I of the section 2 and Fig. 1), the following TCL scripts in OpenSees are run to set up OpenSees as a server:

```
source model.tcl
socket -server accept 7200
vwait forever
```

where "*model.tcl*" is the TCL input file that defines an FE model (refer to step I in Fig. 1. The TCL command "*source*" will run the "*model.tcl*" to set up the model. "*model.tcl*" may also include extra TCL commands that perform analysis for initial loads (e.g., gravity loads). The second command "*socket*" creates a server socket with port number 7200 (it can also be set to other port numbers) and a callback procedure "*accept*" to execute the requested commands whenever the client connects to this server socket and send TCL commands. The third command "*vwait*" sets the server to wait for requests from the client. A simple user-defined TCL function "*accept*" may be defined in server side as

```
proc accept {sock ip port} {
    fconfigure $sock -blocking 1 -buffering none
    fileevent $sock readable [list respond $sock]
   }
```

fconfigure command sets and queries properties of the socket (Welch 2000). The command "*fileevent*" registers a procedure response that is called to provide the real service when the socket is ready for a reading, i.e., after the client finishes sending command to the socket. The user defined procedure "*response*" will provide the service for the client (refer to step IV of the section 2). One example of this "*response*" procedure is

```
proc respond {sock} {
    if {[eof $sock] || [catch {gets $sock data}]} {
        close $sock
    } else {
        eval $data
        global Fx, My
         getTotalResistingForce ; # into Fx, Mx
         puts $sock "$Fx, $My"
        return
     }
     }
```

The command "*gets $sock data*" will get the data from the client via the socket *$sock* (which is usually a string stream including an OpenSees analysis TCL command with parameters) and saves it to the variable *$data*. Then "*eval $data*" will run the command (e.g., run OpenSees one step). The user defined TCL procedure "*getTotalResistingForce*" calculates the total resisting forces (e.g., shear force *$Fx,* and moment *$My*) at the end of the current run, by using the analysis output of OpenSees (e.g., nodal accelerations) which can be accessed by TCL commands (e.g., *nodeAccel*). The above actions are referred to as step IV (1) of the section 2. The command "*puts*" will write these forces to the socket (and thus sends them back to the client, refer to step IV (2) of the section 2).

Client side: The client is a C++ object called "*OpenSeesHandler*" that holds the socket connection from the OpenSees server (see Fig. 1), and is in charge of sending command to OpenSees and getting requested responses back from OpenSees via the socket connection. In the illustration example, Simulink is the other software platform coupled with Opensees. The client ("*OpenSeesHandler*" object) is created as a persistent

object inside a user defined S-function of Simulink at the initialization stage of the Simulink analysis (refer to step II of the section 2). In order to be able to use the TCL library, the user-defined C++ based S-function is compiled and linked together with the TCL library. Thus in the C++ code, it is possible to handle TCL commands. For example, a function *Tcl_Eval*() can be called to run TCL commands in the C++ code. Following TCL scripts are performed to set the connection with OpenSees at the beginning of the process of creating the *"OpenSeesHandler"* object (i.e., in the constructor of the class), which is referred to as step II of the section 2:

```
set s [socket localhost 7200] ;
fconFig.$s -buffering none ;
```

By running this TCL code, *OpenSeesHandler* sets up and holds connection to OpenSees by TCL socket "*$s*" (Note that the same port number 7200 is used in the server side). Then at every analysis step, when it receives a command from Simulink requesting a one-step analysis with parameters such as new acceleration *accel* and the time step *dt*, it will form and run the following TCL commands through socket "$*s*"

```
puts $s "runOpenSeesOneStep 1 $dt $accel "
gets $s
```

The first client side command "*puts $s*" sends the command *"runOpenSeesOneStep 1 $dt $accel"* to the OpenSees server (referred to as step III of the section 2) and wait the response from OpenSees. As already described above, when the OpenSees server receives this command, it calls the server side procedure "*respond*" which performs the following server side TCL codes (refer to step IV of the section 2).

```
eval $data
global Fx, My
getTotalResistingForce ; # into Fx, Mx
puts $sock "$Fx, $My"
```

Note that at the beginning of procedure "*respond*", the command "*gets $sock data"* gets the command *"runOpenSeesOneStep 1 $dt $accel"* and saves it to the variable *"$data"*. After OpenSees finishes the above requested service, it send back the requested responses (i.e., *"$Fx, $My"*), then stops and waits for next call from Simulink. The client side command "*gets $s*" gets "*$Fx, $My*" through the socket *$s* (referred to as part (2) of step IV). The *OpenSeesHandler* thus obtains the responses (*$Fx* and *$My*) which will in turn be used by Simulink to advance the simulation (referred to as step V). The data amount transferred between the client and the OpenSees server is very small for this example (i.e., two strings: *"runOpenSeesOneStep 1 $dt $accel"* and "*$Fx, $My*").

Following is a brief summary of the detailed client side programming work in Simulink (i.e., how to write the user-defined S-function and *OpenSeesHandler* in client side): the persistent object *OpenSeesHandler* is created inside the macro *mdlStart* in S-function. Whenever Simulink asks *OpenSeesHandler* to run OpenSees, it calls *mdlOutputs* macro in its S-function with parameters. This macro then asks *OpenSeesHandler* to form a specific TCL command (e.g., *runOpenSeesOneStep*) and sends it to the OpenSees server (i.e., *puts $s "runOpenSeesOneStep ..."*). Simulink will wait until the responses from the OpenSees server is obtained by *OpenSeesHandler*, and then continue the analysis. Detailed discussion about writing S-functions in Simulink can be found in the literature (Dabney and Harman 2004).