CLIENT-SERVER APPLICATION FOR MULTI-PLATFORM COORDINATION IN REAL-TIME HYBRID SIMULATION TESTING

G. Fermandois[(1)], D. Mera[(2)]

[(1)] *Assistant Profesor, Departamento de Obras Civiles – Universidad Técnica Federico Santa María, Santiago, Chile,* gaston.fermandois@usm.cl

[(2)] *M.S. Student, Departamento de Obras Civiles – Universidad Técnica Federico Santa María, Santiago, Chile,* diego.mera@sansano.usm.cl

Abstract

Real-Time Hybrid Simulation (*RTHS*) is an experimental technique with a physical and computational interface to evaluate complex structural systems. This technique is based on the substructuring methodology in which difficult-to-model components are tested in an experimental setup. Meanwhile, structural components and materials whose behavior can be predicted are modeled numerically. The main feature of RTHS is that all calculations, measurements, and signal processing must be carried out in brief time intervals, typically less than one millisecond. The literature reveals that simple and linear numerical models are used most of the time and are implemented directly in the microcontroller unit for real-time execution. The reason being most research prioritized the study of compensation methods. However, these methods have already reached sufficient maturity, and working with more complex numerical substructures is imperative to achieve realistic results. Thus, it is essential to integrate the capabilities of finite element analysis with the control software used in the experimental setup of *RTHS* tests. Unfortunately, the lack of detailed information on existing integration/coordination methods for *RTHS* testing makes the development and implementation of these coupled systems difficult, especially for researchers who just start working on *RTHS*.

This study's motivation is to provide a reasonable and detailed procedure to achieve the coupling of general FE software with the control software in *RTHS* testing. Computational tasks are interconnected using a client-server communication approach. First, the client is chosen as a Matlab/Simulink program to implement the control algorithms for *RTHS*. Meanwhile, the servers are multiple OpenSees programs that simulate either numerical or experimental substructures. Examples of a virtual *RTHS* benchmark are provided to validate the integration method and illustrate the various uses of the client-server protocol, focusing on earthquake engineering applications. The results show good accuracy and real-time synchronization of client-server communication, allowing for the development of additional features of the Client-Server virtual Real-Time Hybrid Simulation (*CSvRTHS)* framework to model complex structural systems.

Keywords: real-time hybrid simulation; substructuring; communication protocols; client-server; finite element modeling.

1. Introduction

There are different experimental techniques for structural testing in civil engineering, the most modern being hybrid simulation. This technique considers both a physical and computational system interfaced together to evaluate complex systems experimentally [1]. The technique is based on substructuring. The set of structural components and materials whose behavior can be predicted, or those with experimentally calibrated and validated mathematical models, is classified as a numerical substructure. On the other hand, the experimental substructure is a critical component or innovative material without sufficient knowledge to calibrate a model capable of predicting the expected behavior. One technique derived from the conventional hybrid simulation is *Real-Time Hybrid Simulation (RTHS)*. The main feature of *RTHS* is that all calculations, measurements, and signal processing must be performed in very short time intervals, typically less than one millisecond [2]. A critical aspect of *RTHS* testing is that any experimental errors must be mitigated during real-time execution to avoid inaccurate and unstable responses [3]. Therefore, the engineering community has focused on developing compensation techniques in *RTHS*, with acceptable performance and robustness, such that *RTHS* experiments are carried out safely and accurately.

In general, the literature presents *RTHS* tests with simple, linear, numerical substructures models [4–8]. However, if one seeks to achieve greater simulation fidelity, it is desirable to have a finite element package to model the numerical substructure more realistically. Therefore, a fundamental aspect in *RTHS* is integrating a finite element (FE) analysis software with a controller software used in the experimental facility. To date, some programs achieve this goal for hybrid simulation, such as UI-SIMCOR [9], OpenFresco [10] (which is an improved version of the one proposed by [11]), Mercury [12], CSA [13], UT-SIM [14] and the CS technique [15]. Although, their developers no longer support UI-SIMCOR and Mercury. Moreover, the CSA and CS techniques were developed to work on coupling finite element software, not to work with cyber-physical systems. The rest of the coordination programs were originally developed to carry out pseudo-dynamic tests; hence, there are no guarantees for these programs to complete the required calculations deterministically given the time constraints to perform *RTHS* tests. However, the literature provides some examples of conducting *RTHS* with OpenFresco as coordinator software [16]. The problem lies in the small community that supports a continuous and maintained development of OpenFresco, and the lack of detailed documentation for the integration/coordination methods in *RTHS* tests. These issues hinder the development and implementation of *RTHS* tests with high-fidelity coupled systems.

This study's motivation is to provide a reasonable procedure for researchers in *RTHS* to achieve the coupling of widely used FE software with control software available in the laboratory. For this purpose, a four-story steel structure with three bays will be solved, where a frame of the first floor will be considered as the experimental substructure. Virtual environments are previously used for laboratory experimentation as a first approach [17]. Therefore, in this study, a virtual *RTHS* will be carried out using three computer programs. First, Matlab/Simulink [18] provides the most convenient means of implementing control algorithms for *RTHS*. However, Matlab is not the best choice to model structural systems with nonlinear behavior, given that it does not provide a library of elements and materials suitable for structural modeling. Therefore, OpenSees [19] software is preferred for modeling numerical substructures. Finally, the *Open-source Framework for Experimental Setup and Control (OpenFresco)* [10, 20] allows communication from/to OpenSees and a Simulink model running in real-time, and combine a wide variety of hybrid simulation algorithms, control and laboratory systems, experimental setups, and computational simulation models for a specific hybrid simulation.

This paper has the following organization. Section 2 describes the procedure and structure to be solved, the compensation method, and the evaluation criteria to assess its performance. Section 3 presents a detailed description of the communication protocol used to couple OpenSees with Matlab/Simulink applications and the process for executing and evaluating the communication's performance. Section 4 presents the main results obtained from the simulations carried out. Lastly, Section 5 presents the conclusions and final remarks.

2. Problem formulation

2.1 Numerical and experimental substructures

One of the crucial steps when performing hybrid simulation is the correct choice of substructuring [21]. In the context of *RTHS*, instead of solving the entire domain equation of motion, this substructuring process can be applied to subdivide the domain into smaller subdomains so that the order of large and complex structural systems is reduced for computational efficiency. Each subdomain can be resolved independently, as long as the coupling between components is enforced at their interfaces [22].

In this study, a four-story planar frame with three bays is chosen as a reference structure, with a total of 52 degrees of freedom (16 lateral, 16 vertical, and 20 rotational). The beams are modeled with linear elastic frame elements. In contrast, the columns are modeled with frame elements using Giuffre-Menegotto-Pinto steel material with isotropic strain hardening with an initial stiffness of 5.6 $[kip/in]$. Consequently, the first-mode fundamental period is $T_1 = 1.9\ [s]$. The reference structure is divided into the numerical substructure (NS) and experimental substructure (ES), as shown in Fig. 1. One bay of the first story is taken as the ES, while the rest is modeled as NS. Since it is intended to work only with a uniaxial actuator, the boundary degrees of freedom are set to be only horizontal displacements.

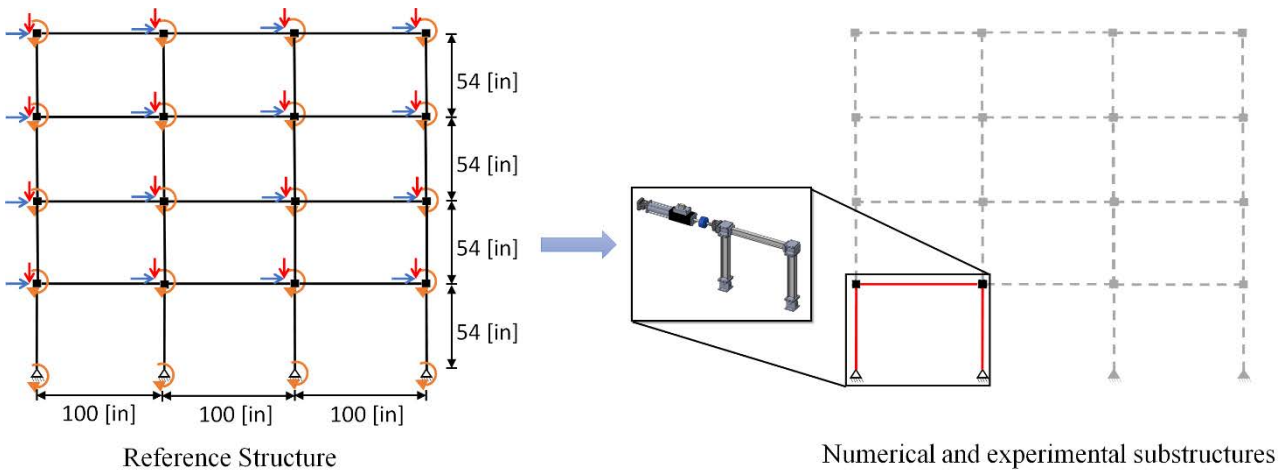


Fig. 1 - Reference structure and substructures. (Note: NS = dashed line; ES = red line)

The equation of motion (EOM) governing the dynamic response of a structure subjected to an input seismic excitation is represented as follows in Eq. (1):

$$M_r\ddot{u} + C_r\dot{u} + F_r(u) = P \tag{1}$$

where M_r and C_r are the mass and damping matrices, respectively. $F_r(u)$ is the nonlinear restoring force. u, $\dot{u}$, and $\ddot{u}$ are the displacement, velocity, and acceleration vectors, respectively, all measured relative to the ground motion. P is the vector of external forces.

Let us now consider the system of two substructures shown previously. We can write the EOM of each substructure as shown below in Eq. (2) and Eq. (3):

$$M^N\ddot{u}^N + C^N\dot{u}^N + F_r{}^N(u^N) = P^N + g^N \tag{2}$$

$$M^E\ddot{u}^E + C^E\dot{u}^E + F_r{}^E(u^E) = P^E + g^E \tag{3}$$

Structural parameters associated with the experimental substructure are indicated with the superscript "E", and those associated with the numerical substructure are indicated with the superscript "N." The terms P^N and P^E are the external force vectors, and terms g^N and g^E are the coupling forces. The main assumption

of this formulation is that the substructures are coupled only through the boundary conditions. Therefore, the coupling forces of the interior degrees of freedom for each substructure must be equal to zero.

For the resolution of this problem, two conditions must be met. First, Eq. (4) will serve as a synchronization indicator on *the boundary degrees of freedom*. Second, Eq. (5) refers to the static equilibrium at the boundary between substructures which should always be enforced.

$$u_b^N = u_b^E \quad \text{(Compatibility)} \tag{4}$$

$$g_b^N + g_b^E = 0 \quad \text{(Equilibrium)} \tag{5}$$

Substituting Eqs. (4) and (5) into Eqs. (2) and (3), the interaction between NS and ES is enforced, which is explicitly shown in Eqs. (6) and (7):

$$M^N \ddot{u}^N + C^N \dot{u}^N + F_r{}^N(u^N) = P^N + \begin{Bmatrix} 0_i^N \\ -g_b^E \end{Bmatrix} \tag{6}$$

$$g^E = \begin{Bmatrix} 0_i^E \\ g_b^E \end{Bmatrix} = M^E \ddot{u}^E + C^E \dot{u}^E + F_r{}^E(u^E) - P^E \tag{7}$$

where $u^N = \{u_i^E \quad u_b^N\}^T$ is the vector of NS displacements, which are classified as internal displacements (indicated by the subscript "i"), and boundary displacements (indicated by the subscript "b"). Eq. (6) is the EOM for NS that is solved ideally using an FEA software. Meanwhile, Eq. (7) collects all terms associated with the force applied from ES to NS at the boundary. This formulation includes all potential effects associated with nonlinear restoring forces, nonlinear damping, and inertial forces, along with any external excitation directly induced to the experimental substructure.

2.2 *CSvRTHS* framework

In *RTHS*, the numerical and experimental substructures (NS and ES, respectively) must communicate in real-time, which is handled by a coordinator software. At each time step of the analysis, the NS solves the equation of motion through numerical integration and produces a target displacement (x_t) at the interface. This signal is imposed on the ES in the laboratory through a loading system (e.g., hydraulic actuator). In *RTHS*, enforcing target displacement must be done in real-time; any delay in communication can cause inaccurate results and potential instability. After this displacement is imposed, the restoring force is measured from the specimen and returned to the NS integrator for the next step.

In this study, a *CSvRTHS* test is implemented, where all components are modeled numerically, including the laboratory test system (actuators, test specimen, and sensors). Both NS, ES, and control systems are modeled with different state-of-the-art software to explore further integration and increased flexibility. Matlab/Simulink is chosen for its numerical capabilities and integration with microcontrollers such as dSpace and Speedgoat. Meanwhile, OpenSees is a finite element analysis (FEA) software widely used by the earthquake engineering community. Each computational task is interconnected using a TCP/IP communication protocol through a client-server (CS) approach [10, 23, 24]. Further details on the CS communication task are provided in Section 3, and the source code is available to the scientific community at a GitHub repository [25].

Many linear and nonlinear models are available for servo-hydraulic actuators [17, 26, 27]. In this study, a model developed by Spencer et al. [27] is used. This model includes mathematical representations of the different components of the test system, including servo valve dynamics, hydraulics, payload, and controller. Fig. 2 shows the model implemented in Simulink. A payload is added to the original model to obtain the actuator displacement, velocity, and acceleration for commanding over to the ES. This payload includes damping and static forces, along with possible force losses based on piston friction. The parameters considered for this model are shown in Table 1.

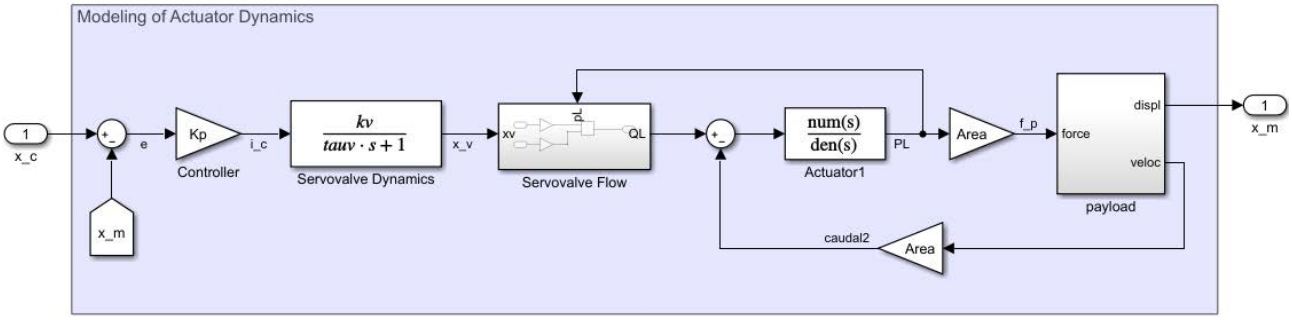


Fig. 2 – Block diagram of actuator dynamics in Simulink.

Table 1 – Actuator model parameters [27].

Component	Parameter	Value	Unit	Description
Controller	K_p	3	mA/in	Proportional gain
Servovalve	τ_v	0.00332	s	Servovalve time constant
	K_q	23.01	$\text{in}^3/\text{s/mA}$	Valve gain
	K_c'	$1.36x10^{-5}$	$\text{in}^3/\text{s/psi}$	Valve flow-pressure gain
Actuator	A	0.751	in^2	Area of the piston
	C_l	$5.89x10^{-6}$	$\text{in}^3/\text{s/psi}$	Leakage coefficient of the actuator
	V_t	48.66	in^3	Total volume of fluid in actuator chambers
	β_e	95958	psi	Effective oil bulk modulus

2.3 Compensation method

The key to *RTHS* is that the commanded displacements are imposed on the test sample by hydraulic actuators in real-time. During testing, the servo-hydraulic actuator's communication, control, and dynamics cause a time delay and lag between components. Time delay/lag in the *RTHS* test is equivalent to introducing spurious negative damping, leading to potential inaccuracies and instabilities.

Therefore, an *Adaptive Model-Based Compensation* (AMBC) [28] is chosen to mitigate time delays during the RTHS tests. The formulation consists of estimating the plant by a third-order transfer function without zeros, as shown in Eq (8).

$$x_m = G_p(s)x_c = \left(\frac{1}{a_3s^3 + a_2s^2 + a_1s + a_0}\right)x_c \tag{9}$$

where x_m is the displacement measured by the sensors. The inverse of this transfer function is used to generate the command signal using the target displacement as input, as shown in Eq (10). The derivatives of x_t are obtained with finite differences.

$$x_c = G_p^{-1}x_t = (a_3s^3 + a_2s^2 + a_1s + a_0)x_t = a_3\dddot{x}_t + a_2\ddot{x}_t + a_1\dot{x}_t + a_0x_t \tag{10}$$

The adaptive parameters a_i, $i = 0,1,2,3$, must be adjusted during the test to achieve good compensation. This adaptation is formulated through a differential equation model provided in Eq. (11), where $x_m^{(i)}$ is the i-th derivative of x_m associated with adaptive parameter a_i. A Butterworth filter is designed to remove high-frequency noise and to estimate higher-order derivatives of target and measured signals.

$$\dot{a}_i = \Gamma_i\left(\frac{x_c - [a_3\ a_2\ a_1\ a_0][\dddot{x}_m\ \ddot{x}_m\ \dot{x}_m\ x_m]^T}{1 + [\dddot{x}_m\ \ddot{x}_m\ \dot{x}_m\ x_m][\dddot{x}_m\ \ddot{x}_m\ \dot{x}_m\ x_m]^T}\right)x_m^{(i)} \tag{11}$$

This methodology requires the calibration of adaptive gains. For this, the procedure given in the original paper [28] is followed, but with certain modifications: (i) ground acceleration scaled to 100%; (ii) multiples single-degree-of-freedom structures with a natural period between $0.6 - 4\,[s]$; and (iii) mass, stiffness, damping, and initial conditions for the adaptive controller were randomly generated through the Latin hypercube method to cover a larger range of structures, obtaining more robust gains. The calibration of optimal adaptive gains consists of a global optimization scheme through particle swarm optimization, which minimizes the performance indicator J_2 explained in Section 2.3. Additionally, the initial conditions are defined from the actuator without a specimen attached. Finally, the values obtained and used for the adaptive gains (Γ_i) and initial conditions (a_i) are: $\Gamma_i = \mathrm{diag}(10^{6.03} \quad 10^{4.23} \quad 10^{1.48} \quad 10^{-1.10})$; $a_i = [1.0 \quad 1.99 \cdot 10^{-2} \quad 10^{-4} \quad 0.0]$.

2.4 Evaluation criteria

The performance of the *CSvRTHS* is evaluated through three performance indicators, which are shown in Table 2. J_2 is the normalized root-mean-square error between the target displacement (x_t) and the measured displacement (x_m); this indicator measures the synchronization error. J_4 is the normalized root-mean-square error between the reference displacement (x_r) and the measured displacement (x_m), which measures the error between the simulation and the reference structure. Both J_2 and J_4 indices were provided in Silva et al. [17]. Also, τ corresponds to the delay indicator obtained with the frequency evaluation index [29], which measures the synchronization delay (in milliseconds) between x_t and x_m. Finally, MT is the missed ticks of the real-time clock in Simulink Desktop Real-Time, which will be discussed in the next section.

Table 2 - Performance evaluation criteria.

Criterion	Description	Definition	Units
J_2	Normalized root mean square of the tracking error	$J_2 = \sqrt{\frac{\sum_{i=1}^{N}[x_m(i) - x_t(i)]^2}{\sum_{i=1}^{N}[x_t(i)]^2}} \times 100$	%
J_4	Normalized RMS relative displacement at the first floor	$J_4 = \sqrt{\frac{\sum_{i=1}^{N}[x_m(i) - x_r(i)]^2}{\sum_{i=1}^{N}[x_r(i)]^2}} \times 100$	%
τ	Time delay between the input and output signals evaluated at the equivalent frequency.	$\tau = -\frac{\phi_0}{2\pi f^{eq}}$	ms
MT	Missed real-time clock ticks and their associated data points.	Internally calculated by Simulink	ticks

3. Methodology

3.1 Communication protocol

In the Client-Server (CS) communication protocol, tasks or workloads are divided between providers of a resource or service, called servers, and service requesters called clients [23]. This architecture is ideal for communicating of the NS and ES as servers with Simulink as the client. The whole process is better explained in Fig. 3. In general terms, it starts with the NS implemented in OpenSees (server), where it calculates the displacement in the interface nodes and sends this signal to Simulink (client). In Simulink, this displacement is received, and after a series of intermediate steps (multi-rate transitioning, compensation method, actuator dynamics and control), it is sent to the ES (server). The ES given a displacement returns a force to Simulink, to finally return that force to the NS. With this information, the NS performs the calculations of the next time step.

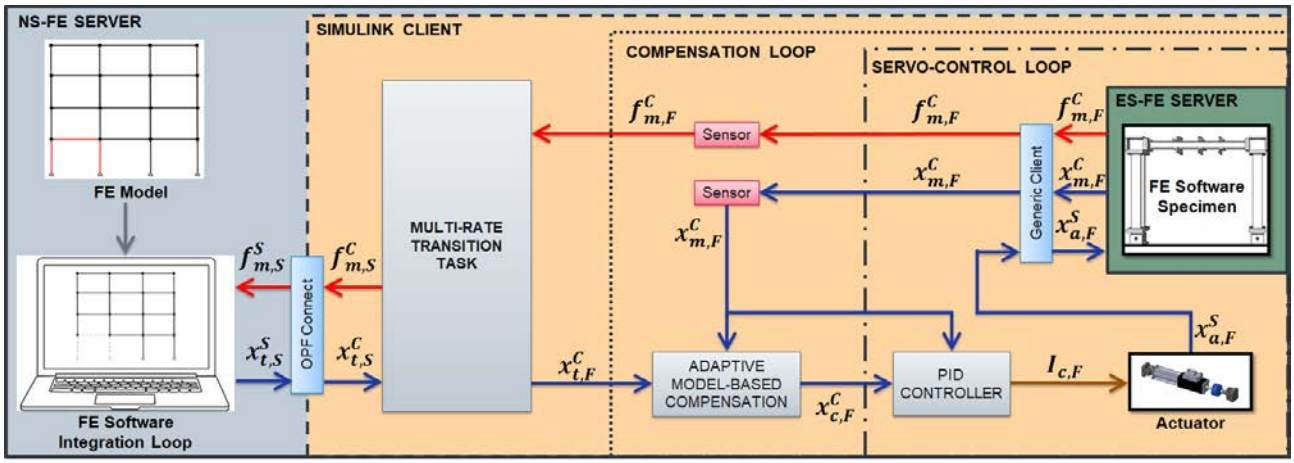
It is worth noting that both OpenSees and Matlab/Simulink supports TCP/IP communication protocol. But it requires a certain level of programming knowledge to implement this interface. Although it is possible

to easily open and close communication channels through Matlab scripts [23], it is not recommended for *RTHS* since it will cause severe time constraints during Simulink client's real-time execution. An alternative employs user-defined blocks called S-Functions [24] in Simulink, which can execute custom code using different programming languages and include a set of callback methods that perform the necessary tasks at each time step. During the simulation stage, the Simulink engine calls the appropriate methods for each S-function block in the model. For this study, a Level 2 S-function written in C language is suitable to implement TCP/IP communication. This S-Function block is compiled into a Matlab Executable (MEX) file, which is necessary to run user-defined code in Matlab/Simulink.

In summary, a Level 2 S-Function called *Generic-Client* is used to communicate with the ES. This user-defined block interfaces the Simulink (client) with the ES by specifying IP and connection port as parameters. As input data, only this displacement is required, and the server returns a displacement and a force. On the other hand, the Level 2 S-Function called *OPF Connect* is used to interface Simulink with the NS, in which force and displacement are sent to solve the equation of motion, Eq. (6), and it responds with a displacement in the interface nodes. Both user-defined blocks can be obtained from OpenFresco Version 2.7.1 source code [30] and readily available for use in this study's companion GitHub repository [25].

3.2 Implementation of real-time system

The implementation of *CSvRTHS* is illustrated in Fig. 3. Detailed instructions of software installation and execution are provided in the companion GitHub repository [25]. Briefly, the first requirement is to start the ES-FE server, which is waiting for the simulation to start. Then, Simulink client must be executed, which requests a target displacement to OpenSees to execute of the first temporary step. Finally, the NS-FE server containing the NS performs numerical integration and begins the simulation. An explicit integration scheme, such as the central difference method (NS-FE server) and the Dormand-Prince method (Simulink client). The sampling intervals used for the fast (client and ES-FE server) and slow (NS-FE server) processes are $1/1024$ and $2/100$ [s], respectively (i.e., $0 < T_{Fast} < T_{Slow}$). To summarize, the loops need to be run from the inside to the outside layers: (1) ES-FE server (green); (2) Simulink client (orange); and (3) the NS-FE server (blue-gray).



Note. Signals: x = displacement, f = force; superscripts: ^{C} = client, ^{S} = server; subscripts: $_m$ = measured, $_t$ = target, $_c$ = compensated, $_a$ = actuator, $_S$ = slow rate, $_F$ = fast rate)

Fig. 3 – Simulation loops considered in *CSvRTHS*.

Further, a real-time sync block is placed on the system to synchronize the Simulink model with the real-time kernel clock and ensure that it does not run faster than the user-specified simulation rate. Using this sync block implies that at each sample interval, the Simulink model is evaluated in real-time. Simulink writes the input data into a buffer that it passes to the kernel-mode process. The kernel-mode process propagates the data to the hardware, which writes response data into another buffer. Simulink reads the response data at the next

time step and propagates it to the rest of the model. In this case, the Accelerator mode of Simulink Desktop Real-Time is used. This mode implies that simulation is not a "hard" *RTHS*; that is, producing results after their deadline is allowed and does not cause catastrophic consequences on the system under control.

A consequence of this limited synchronization is that the simulation can miss consecutive real-time clock ticks and their associated data points without failure [31]. The reasons why ticks are lost are various; for example, model complexity implies that the model can be so complex that Simulink cannot keep up with the kernel in real-time. In this paper, the number of allowable missed ticks (MT) was fixed to 500. Consequently, the number of missed ticks is selected as indicators of the communication protocol's performance in the *CSvRTHS* framework. The reliability of this real-time system is assessed in Sec. 4.2.

3.3 Multi-rate transitioning

One problem working with coupling programs is that the systems may run at different speeds. In this case, El Centro 1940 is selected as the seismic record, which implies that OpenSees will perform the integration in time steps of 0.02 $[s]$, corresponding to the step presented by the record. On the other hand, a time step of $1/1024$ $[s]$ is selected to perform the control algorithms in Simulink. For multi-rate synchronization, the literature provides multiple alternatives [32–34]. In this study, the corrector-predictor algorithm proposed by Nakashima [35] is selected, which for the continuous generation of signals uses polynomial extrapolation/interpolation. This algorithm is implemented in a StateFlow block in Simulink and can be obtained in the OpenFresco source code [30]. Fig. 4 shows the displacements from OpenSees, which have a staircase pattern given their greater temporal distribution. The target displacement obtained from the corrective predictor algorithm is observed to smooth this signal for the faster Simulink task.

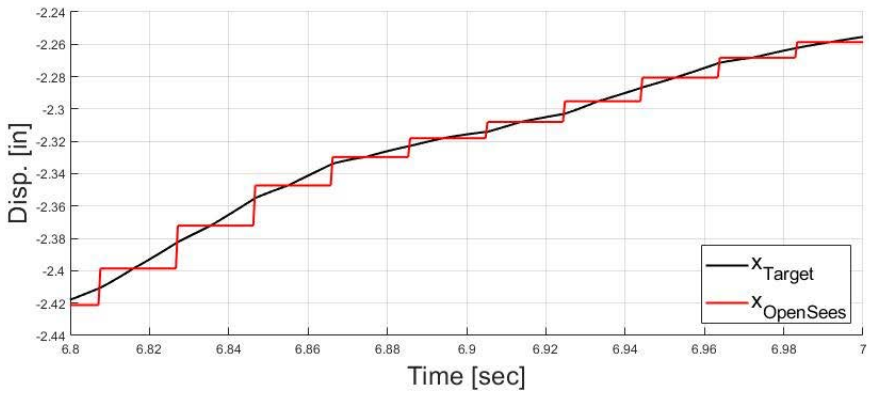


Fig. 4 – Predictor-corrector response in *CSvRTHS*.

4. Results

4.1 CSvRTHS performance

The *CSvRTHS* results are presented graphically in this subsection. For displacement comparisons, the node to the left of the first floor (upper left node of the ES) is selected as representative. In Fig. 5, the measured displacement is compared to the target displacement for evaluating the compensation method. With $J_2 = 0.866$ [%] and $\tau = 0.0075$ [msec], a good tracking is achieved, which allows a stable test. Furthermore, it is noticeable that practically all the error obtained (J_2) is due to the simulated measurement noise considered in the sensors.

Meanwhile, Fig. 6 shows a comparison between the measured displacement and the reference displacement. The J_4 indicator reaches a value of 2.92 [%], which means that correct and accurate results were obtained. The reference displacement is obtained from a model made entirely in the OpenSees software.

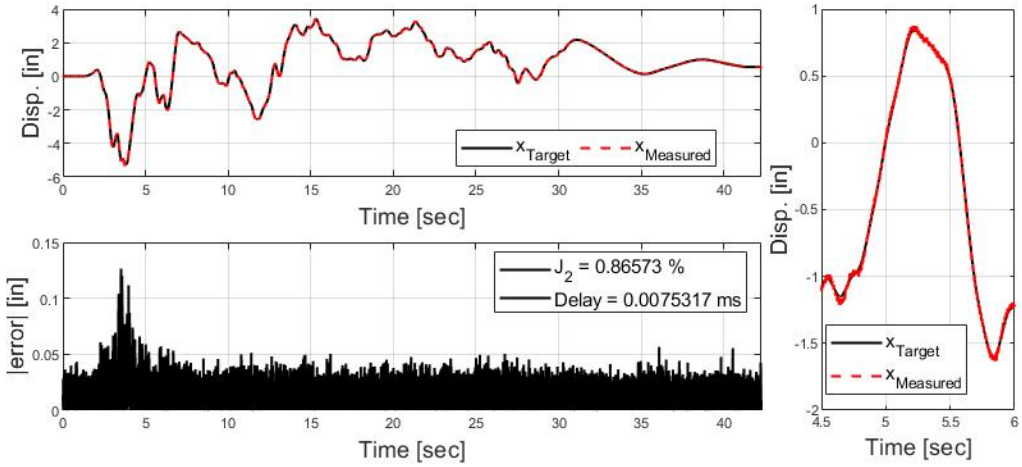


Fig. 5- Target v/s Measured Comparison and J2 indicator.

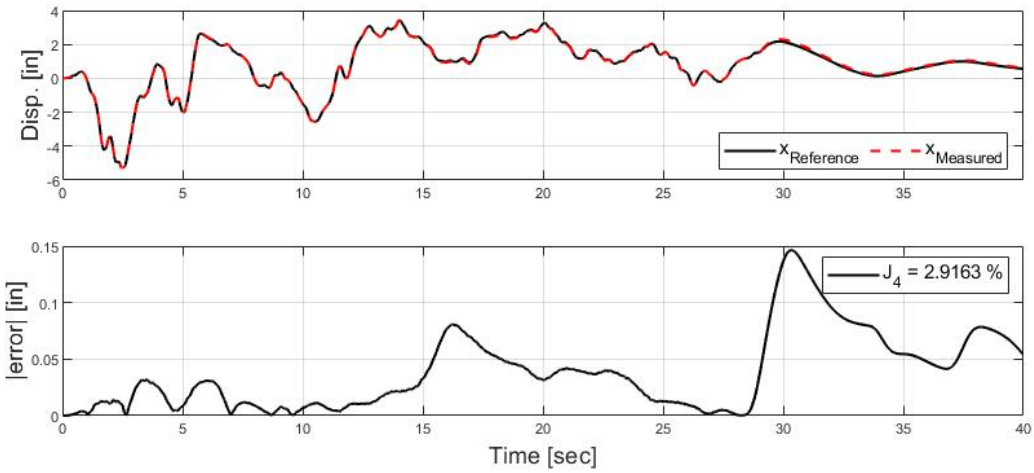


Fig. 6 - Reference v/s Measured Comparison and J4 indicator.

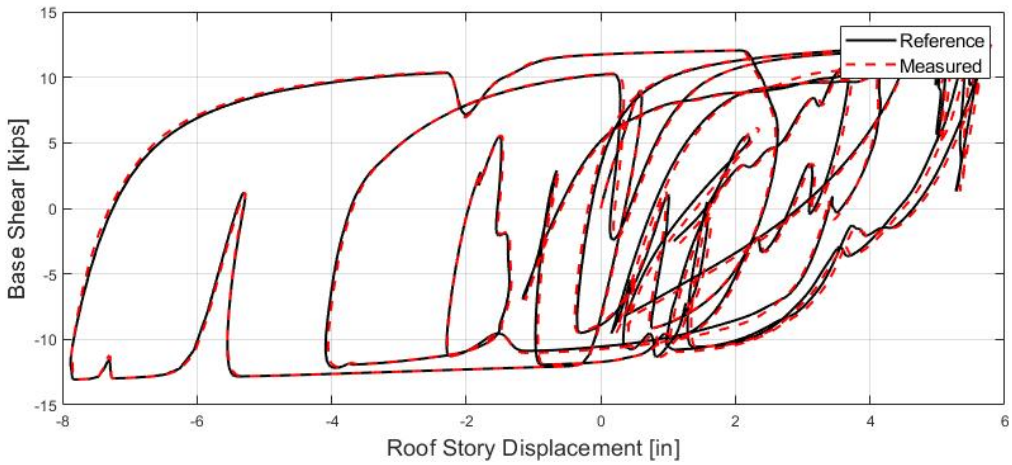


Fig. 7 - Global comparison. Base Shear v/s Roof Story Displacement.

Finally, a comparison in global terms of the structure is added in Fig. 7. For this purpose, the graph of roof story displacement versus base shear is made. Again, a good concordance of the measured results against the reference ones is observed, thus validating the substructuring and compensation methodology.

4.2 Reliability of the real-time system

The reliability of *CSvRTHS* is assessed through a set of 40 simulation runs. The simulations were carried out on an MSI laptop, model GF75 Thin 9RCX (2.6 GHz Intel Core i7-9750H processor, 8 Gb RAM), running Windows 10 version 20H2, running Simulink Desktop Real-Time in Accelerator Mode, and under controlled room temperature (19°C). The results are shown in Fig. 8, which presents missed ticks statistics during the real-time execution. The maximum value of missed ticks is 426, which is the worst-case scenario. Therefore, the limit of 500 (established in Section 3.2) is never exceeded, and a fast and efficient communication protocol is achieved. It is observed that the missed ticks are concentrated in the first 3 to 4 [s]. Hence, the seismic record is zero-padded for the first 5[s] to verify that the excitation does not cause this phenomenon. This communication overhead is probably caused by the initiation of the communication protocol between applications.

For steady-state real-time simulation (i.e., times after 4 [s]), the missed tick statistics are significantly reduced. The lower right graph of Fig. 8 shows a worst-case scenario of 18 missed ticks and a zero median. The peaks of missed ticks are generated in regular intervals of 0.02 [s], coinciding with the duration of the slow rate process (NS-FE Server). However, it is a low and stable number of missed ticks, compared to the previous peaks, so it is assumed a greater precision and accuracy in the results after the 4 [s] start of the *CSvRTHS*. This result encourages using a seismic record with 5 [s] of zero-padding at the start, thus ensuring that critical information is not lost.

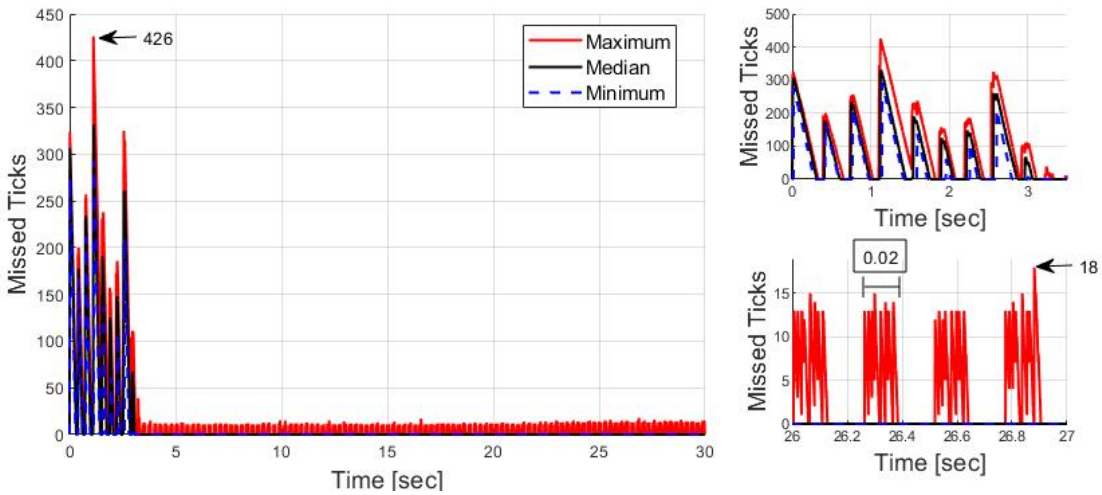


Fig. 8 – Statistics of the missed ticks during *CSvRTHS* simulation.

Another simulation alternative is Simulink External Mode, where the resulting executables run in the operating system kernel mode of the development computer. Parameter data is exchanged with Simulink via a shared memory interface, allowing full synchronization with the real-time clock. Consequently, missed ticks are eradicated in this simulation mode, allowing for the development of a "hard" real-time system. Further investigation of this topic will be carried out in future studies.

5. Conclusions

This study presents a detailed explanation of the fundamental components of *CSvRTHS*, a virtual real-time hybrid simulation implemented with a Client-Server communication protocol for coupling a finite element analysis software (OpenSees) with the real-time control system (Matlab/Simulink). A structural engineering

example is provided to illustrate its implementation and software validation. Additionally, a previously developed adaptive compensation method is adopted and validated against a nonlinear specimen, demonstrating that a robust fixed controller could be used with different experimental substructures, avoiding subsequent system identification tests.

Validation of the CS communication protocol is achieved for *vRTHS*, providing enough evidence of this protocol's modularity and flexibility to connect different computational tasks. Further research will consider implementing the CS method in *RTHS* over a network with dedicated microcontroller (client) and FEA machines (servers) in a laboratory setup. The authors anticipate that this work will boost the development and deployment of real-time hybrid simulation with high-fidelity numerical models and promote further collaboration among experts in the field.

6. Acknowledgments

The authors gratefully acknowledge the financial support from *Agencia Nacional de Investigación y Desarrollo* (ANID, Chile) through FONDECYT Iniciación research project No. 11190774. Also, we thank Cristobal Gálmez, M.S. Student at USM, for his assistance with the calibration of the Adaptive Model-Based Compensation and provide insight and expertise with *RTHS* that greatly assisted this research. Finally, any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect those of the sponsors.

7. References

[1] M. Nakashima, "Hybrid simulation: An early history," *Earthq. Eng. Struct. Dyn.*, no. March, pp. 1–14, Apr. 2020.
[2] M. Nakashima, "Development, potential, and limitations of real-time online (pseudo-dynamic) testing," *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 359, no. 1786, pp. 1851–1867, Sep. 2001.
[3] T. Horiuchi, M. Inoue, T. Konno, and Y. Namita, "Real-time hybrid experimental system with actuator delay compensation and its application to a piping system with energy absorber," *Earthq. Eng. Struct. Dyn.*, vol. 28, no. 10, pp. 1121–1141, Oct. 1999.
[4] A. Ashasi-Sorkhabi, H. Malekghasemi, and O. Mercan, "Implementation and verification of real-time hybrid simulation (RTHS) using a shake table for research and education," *JVC/Journal Vib. Control*, vol. 21, no. 8, pp. 1459–1472, 2015.
[5] R. Enokida, "Basic examination of two substructuring schemes for shake table tests," *Struct. Control Heal. Monit.*, vol. 27, no. 4, pp. 1–23, 2020.
[6] Z. Fei, W. Jinting, J. Feng, and L. Liqiao, "Control performance comparison between tuned liquid damper and tuned liquid column damper using real-time hybrid simulation," *Earthq. Eng. Eng. Vib.*, vol. 18, no. 3, pp. 695–701, 2019.
[7] T. L. Karavasilis, J. M. Ricles, R. Sause, and C. Chen, "Experimental evaluation of the seismic performance of steel MRFs with compressed elastomer dampers using large-scale real-time hybrid simulation," *Eng. Struct.*, vol. 33, no. 6, pp. 1859–1869, 2011.
[8] A. Stavridis and P. B. Shing, "Hybrid testing and modeling of a suspended zipper steel frame," *Earthq. Eng. Struct. Dyn.*, vol. 39, no. 2, pp. 187–209, 2010.
[9] O.-S. Kwon, A. S. Elnashai, B. F. Spencer Jr., and K. Park, "UI-SIMCOR : A global platform for hybrid distributed simulation," in *9th Canadian Conference on Earthquake Engineering*, 2007, no. June, pp. 139–149.
[10] A. H. Schellenberg, S. A. Mahin, and G. L. Fenves, "Advanced Implementation of Hybrid Simulation. PEER Report 2009/104," *Peer 2009/104*, no. November, p. 286, 2009.
[11] Y. Takahashi and G. L. Fenves, "Software framework for distributed experimental-computational simulation of structural systems," *Earthq. Eng. Struct. Dyn.*, vol. 35, no. 3, pp. 267–291, Mar. 2006.
[12] V. Saouma, D.-H. Kang, and G. Haussmann, "A computational finite-element program for hybrid simulation," *Earthq. Eng. Struct. Dyn.*, no. 41, pp. 375–389, 2012.
[13] Y. Zhang, P. Pan, Q. Gu, J. Yang, and K. Deng, "Development of Collaborative Structure Analysis (CSA) system and its application to investigate effects of soil-structure interaction," *J. Earthq. Eng.*, vol. 18, no. 7, pp. 1151–1169, 2014.
[14] P. Mortazavi, X. Huang, O. Kwon, and C. Christopoulos, "An Overview of the University of Toronto Simulation (UT-SIM) Framework and its Application to the Performance Assessment of Structures," in *7th International*

Conference on Advances in Experimental Structural Engineering, 2017.
[15] Q. Gu and O. Ozcelik, "Integrating OpenSees with other software - with application to coupling problems in civil engineering," *Struct. Eng. Mech.*, vol. 40, no. 1, pp. 85–103, Oct. 2011.
[16] T. Li, M. Su, Y. Sui, and L. Ma, "Real-time hybrid simulation on high strength steel frame with Y-shaped eccentric braces," *Eng. Struct.*, vol. 226, no. September 2020, p. 111369, Jan. 2021.
[17] D. Gomez, C. E. Silva, D. Gomez, A. Maghareh, and S. J. Dyke, "Benchmark control problem for real-time hybrid simulation Benchmark control problem for real-time hybrid simulation," *Mech. Syst. Signal Process.*, vol. 135:106381, no. October, 2019.
[18] Matlab, *Version 9.9.0 (R2020b)*. Natick, Massachusetts: The MathWorks Inc., 2020.
[19] F. McKenna, "OpenSees: A framework for earthquake engineering simulation," *Comput. Sci. Eng.*, vol. 13, no. 4, pp. 58–66, 2011.
[20] A. Schellenberg, H. K. Kim, Y. Takahashi, G. L. Fenves, and S. A. Mahin, "OpenFresco Command Language Manual," no. July, 2009.
[21] S. N. Dermitzakis and S. A. Mahin, "Development of Substructuring Techniques for On-Line Computer Controlled Seismic Performance Testing," Berkeley, CA, UCB/EERC-85/04, 1985.
[22] D. de Klerk, D. J. Rixen, and S. N. Voormeeren, "General Framework for Dynamic Substructuring: History, Review and Classification of Techniques," *AIAA J.*, vol. 46, no. 5, pp. 1169–1181, May 2008.
[23] M. Sysel, "MATLAB/Simulink TCP/IP Communication," in *Proceedings of the 15th WSEAS international conference on Computers*, 2015, no. July 2011.
[24] Matlab, *Simulink, Model-Based and System-Based Design: Writing S-Functions Version 5*. Natick, Massachusetts: The MathWorks Inc., 2002.
[25] D. Mera and G. Fermandois, "Client-Server application for vRTHS," 2021. [Online]. Available: https://github.com/FermandoisLab/CSvRTHS.
[26] A. Maghareh, C. E. Silva, and S. J. Dyke, "Parametric model of servo-hydraulic actuator coupled with a nonlinear system: Experimental validation," *Mech. Syst. Signal Process.*, vol. 104, pp. 663–672, 2018.
[27] J. E. Carrion and B. F. Spencer, "Model-based Strategies for Real-time Hybrid Testing. Technical Report NSEL-006," Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 2007.
[28] G. Fermandois, C. Galmez, and M. Valdebenito, "Optimal Gain Calibration of Adaptive Model-Based Compensation for Real-Time Hybrid Simulation Testing," *17th World Conf. Earthq. Eng. (17WCEE), Sendai, Japan*, 2020.
[29] T. Guo, C. Chen, W. Xu, and F. Sanchez, "A frequency response analysis approach for quantitative assessment of actuator tracking for real-time hybrid simulation," *Smart Mater. Struct.*, vol. 23, no. 4, p. 045042, Apr. 2014.
[30] A. H. Schellenberg, "OpenFresco," *GitHub Repository*, 2018. [Online]. Available: https://github.com/aschellenberg74/OpenFresco.
[31] *Simulink Desktop Real-Time User's Guide*. Natick, Massachusetts: The MathWorks Inc., 2020.
[32] A. Maghareh, J. P. Waldbjørn, S. J. Dyke, A. Prakash, and A. I. Ozdagli, "Adaptive multi-rate interface: development and experimental verification for real-time hybrid simulation," *Earthq. Eng. Struct. Dyn.*, vol. 45, no. 9, pp. 1411–1425, Jul. 2016.
[33] Y. Zhang, R. Sause, J. M. Ricles, and C. J. Naito, "Modified predictor-corrector numerical scheme for real-time pseudo dynamic tests using state-space formulation," *Earthquake Engineering and Structural Dynamics*, vol. 34, no. 3. pp. 271–288, 2005.
[34] J. Ghaboussi, G. J. Yun, and Y. M. A. Hashash, "A novel predictor-corrector algorithm for sub-structure pseudo-dynamic testing," *Earthq. Eng. Struct. Dyn.*, vol. 35, no. 4, pp. 453–476, 2006.
[35] M. Nakashima and N. Masaoka, "Real-time on-line test for MDOF systems," *Earthq. Eng. Struct. Dyn.*, vol. 28, no. 4, pp. 393–420, Apr. 1999.

