HyTest: Platform for Structural Hybrid Simulations with Finite Element Model Updating

Ge Yang, Bin Wu, Ge Ou, Zhen Wang, Shirley Dyke

Abstract

Hybrid simulation has been demonstrated to be a powerful method to evaluate the system-level dynamic performance of structure. With the numerical substructure analyzed with finite element software and the difficult-to-model components tested with an experimental substructure, complex structures with sophisticated behaviors can readily be examined through a hybrid simulation. To coordinate and synchronize the substructures in hybrid simulation, software is required. In recent studies, model updating has been integrated into hybrid simulation to improve testing accuracy by updating the numerical model during the analysis. However, online model updating scheme requires some modifications in the typical hybrid simulation testing procedure, and this greater complexity is entailed in its implementation regarding the collaboration of identification algorithms with existing hybrid simulation software. To address this issue and broaden the utilization of hybrid simulation with model updating, an existing platform named HyTest originally for conventional hybrid simulation is extended for this purpose. This version of HyTest facilitates the online identification of material constitutive parameters using experimental measurements in its finite element based identification module. It also includes a data center with a uniform data transmission protocol to incorporate different substructures and modules. A numerical example is used to demonstrate the online identification of material parameters for concrete and steel models in a reinforced column, and to verify the accuracy of the identification module. Lastly the effectiveness of HyTest in conducting hybrid simulation with model updating is validated using actual hybrid tests on a steel frame.

1. Introduction

Hybrid simulation in civil engineering originated from pseudo-dynamic testing for structures subjected to seismic excitations [1,2], in which the static reaction force of a structure is measured from the specimen, while the inertia and damping forces are calculated in an online computer. Although the pseudo-dynamic testing reduces the test cost by driving the structural model in a quasi-static way instead of dynamic loading, it is usually economically prohibitive to physically test the entire structure for large-scale ones. By physically testing only the critical or complex components, while analytically simulating the rest of the structure, hybrid simulation can capture realistic and detailed behavior of tested components, as well as provide a system-level performance of the entire structure [3,4]. In most previous applications of hybrid simulation, the adopted numerical models were assumed to be linear [5], or accurate enough to describe nonlinearity of the numerical substructure [6–8]. However, the assumption limits the application of hybrid simulation to more sophisticated structures which contain several nonlinear components. Due to limited resources of testing facilities, it is impossible to take all the uncertain parts as the experimental substructure. Therefore the accuracy of the numerical substructure cannot be guaranteed. Moreover, some of the uncertain components may have similar nonlinear behavior, typical of which are the buckling restrained braces placed in the same building. In this case, physically testing all these components would be unnecessarily costly even with enough laboratory facilities.

Hence, an online model updating scheme has been introduced into hybrid simulation [9,10] to achieve better accuracy. In this approach, the component expected to experience the largest
deformation among all the similar components is represented by the experimental substructure, while the remainder is modeled with adjustable parameters as the numerical substructure. Once the experimental substructure yields and provides sufficient information for describing nonlinearity, these adjustable parameters are calibrated and updated incrementally based on measured information. Online model updating hybrid simulation, also called as hybrid simulation with model updating (HSMU), has been validated for different applications in recent years. Kwon and Kammula [11] assumed that the restoring force of the numerical substructure could be represented by a weighted average of several candidate numerical models with possible parameter variation. The weighting factor of each model was identified and updated in their study. Yang et al. [12] showed the potential of updating constitutive parameters at the element level through a numerical study, which was later explored experimentally by both quasi-static hybrid simulation [13] and real-time hybrid simulation [14–16]. To extend model updating to more sophisticated numerical models, researchers also employed constitutive parameter updating at the sectional level [17] and material level [18,19].

Despite the advantage of online model updating to improve the accuracy of hybrid simulation, there is added complexity in the implementation. Three major modifications to the conventional hybrid simulation procedure are required. First, the model parameters of the numerical substructure need to be updated online during the simulation. There is a growing trend in hybrid simulation that numerical substructures are analyzed with finite element software. However, finite element software is not originally designed for updating model parameters. Hence, the adopted software needs to be modified to enable this feature. Second, added data transmission is required concerning experimental information and updated parameters for the identification procedure. Finally, the identification process itself may involve interacting with finite element software, to use more realistic structural models describing experimental substructures for identification. To realize the full potential of HSMU to utilize sophisticated finite element software, a platform for HSMU should address all these three issues.

Currently some well-established platforms exist for conducting conventional hybrid simulation. One of the most widely used software for hybrid simulation is OpenFresco, developed at the University of California, Berkeley [20]. OpenFresco is a middleware to help standardize the employment of both local and distributed hybrid simulation. It has a three-tier architecture, in which the top tier represents finite element software, the backend tier handles control and data acquisition systems, and the middle tier serves as the bridge between these two tiers. OpenFresco is able to readily interact with OpenSees [21], as they were originally designed within the same object-oriented framework. It can communicate with a wide variety of control and data acquisition systems such as MTS, dSPACE, LabVIEW, SCRAMNet and xPCTarget, and computational software such as ABAQUS [22], LS-Dyna [23], and ZeusNL [24].

Another well-known software for hybrid simulation is UI-SimCor, developed at the University of Illinois at Urbana-Champaign [25]. This framework is intended to support multi-component hybrid simulation, with each component being represented by a shadow module in UI-SimCor, and communicating with a static analytical tool or an experimental system through TCP/IP, LabVIEW or NHCP (NEES Hybrid Communication Protocol). It supports various finite element software, such as ZeusNL, OpenSees, ABAQUS, and FedeasLab [26]. As UI-SimCor is developed with MATLAB, it can also communicate with OpenFresco through TCP/IP coded in a MEX file.

Pan et al. [27] proposed a system known as peer-to-peer (P2P) internet online hybrid test system, which is intended for distributed hybrid simulation. In this system, each substructure is equally treated, with a ‘Coordinator’ handling the equilibrium and compatibility of the boundaries between substructures. Instead of building an overall dynamic model, the equation of motion is formulated by each substructure separately, and can be considered in existing finite element software such as OpenSees and ABAQUS. Other examples of software platforms include ISEE [28], NetSlab [29], and HyTest [30].

These platforms have been extensively used in the research community to conduct hybrid simulation. However, as they are originally designed for conventional hybrid simulation with constant model parameters, extra efforts may be needed for implementing HSMU using these existing platforms. Elanwar et al. [19] modified the source code of UI-SimCor and ZeusNL to exchange information with an identification algorithm. Four types of material models in ZeusNL (‘STL1’, ‘STL2’, ‘STL3’ for steel, and ‘CON2’ for concrete) were modified to enable parameter updating. However, this modification was made for specific cases in which the stress of material was assumed to be measured experimentally.

For facilitating the deployment of HSMU, the existing hybrid simulation software platform called HyTest is extended by incorporating modules for model identification and updating. The new version of HyTest provides a data center with a uniform data transmission protocol which can be adopted by substructures as well as the identification module to collaborate with each other. Moreover, the identification module of HyTest can employ OpenSees to conduct finite element model identification, which enables more realistic and sophisticated structural models. This paper will describe the main features of HyTest with model updating, and present relevant numerical and experimental verifications.

2. Hybrid Simulation with Finite Element Model Updating

Model updating, when introduced into hybrid simulation, aims to improve the accuracy of the numerical substructure, and thus of the entire hybrid simulation. The approach involves two steps. First, parameter identification is performed based on experimental information, and second these parameters are used to update the numerical substructure. Figure 1 shows the schematic of a typical hybrid simulation with model updating. The prototype structure is a simple three degree-of-freedom, three-story shear-type steel frame, subjected to a lateral seismic excitation. The first story, which is expected to experience higher nonlinearity, is taken as the

![Fig. 1. Schematic of a typical hybrid simulation with model updating.](Image 330x644 to 374x719)
Table 1
Design objectives and strategies of HyTest.

<table>
<thead>
<tr>
<th>Design objective</th>
<th>Implementation strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capability to interact with analytical software and control system</td>
<td>Interfaces for communication with OpenSees, ABAQUS, MTS control system, and NI system.</td>
</tr>
<tr>
<td>Conducting HSMU</td>
<td>Interacting with MATLAB to use identification tools, and integrating OpenSees to build and update numerical model.</td>
</tr>
<tr>
<td>Ease of involving other methods and software</td>
<td>Data center-client strategy with uniform data transmission protocol; similar procedure for collaborating with different software.</td>
</tr>
<tr>
<td>User-friendly interface for operation</td>
<td>In the fundamental modules for HSMU such as time integration algorithm, boundary coordination, iteration algorithm, and substructure simulation, the most commonly-used methods are included and visualized.</td>
</tr>
</tbody>
</table>

The coupled equation of motion of the entire structure is

\[ M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + R_N(\mathbf{x}_N, \theta) + R_E(\mathbf{x}_E) = -M\ddot{\mathbf{x}}_S \]  

where subscripts ‘N’ and ‘E’ indicate the numerical substructure and the experimental substructure, respectively; \( M \) and \( C \) are the mass and damping matrices of the entire structure; \( \mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}} \) are the vectors of overall displacement, velocity, and acceleration, respectively; \( R \) represents vectors of restoring force; \( \theta \) is the parametric vector identified from the experimental substructure.

Unlike traditional model updating in the field of system identification, which commonly updates the numerical model of a specimen itself, here the identified model parameters are used to update the numerical substructure in HSMU. For successful use of model updating in hybrid simulation, the experimental substructure and the numerical substructure should share the same parameters to be identified and updated. Moreover, for better conditioning, the number of parameters should be kept small, and engineering insight is needed to provide reasonable initial parameter values [31]. Thus, the parameters of material constitutive model are a good choice to be identified. On the one hand, the numerical component may not be exactly the same as the experimental substructure, such as the geometric length, the sectional shape, but they do share the same material properties. One the other hand, with prior material coupon test, or just ordinary engineering experiences, the initial estimations of material parameters are relatively accurate with reduced uncertainties. Hence, the parameters of material constitutive relationship are chosen to be identified in this paper.

In the identification procedure, experimental displacements \( \mathbf{x}_E \) and measured restoring forces \( \mathbf{R}_E \) are adopted as the experimental information needed for identification. A finite element model of the experimental substructure is built to implicitly formulate the relationship between material parameters and restoring forces.

3. HyTest System Development

The design of new version of HyTest aims to offer a tool for hybrid simulation using finite element model updating. Table 1 provides a general view on the design objectives as well as the implementation details that address them specifically.

3.1. Data center-client architecture

Considering these four objectives, HyTest is designed to use multiple clients dealing with the time integration, identification part, numerical and experimental substructures separately. The schematic interactions between each client are demonstrated in Figure 2, which represents the typical data transmission flow of an HSMU. For HyTest, a data center-client architecture is utilized. The data center serves as a middle tool for transmission between different modules while each client runs independently without direct interaction with other clients. Figure 3 provides an architectural overview of the proposed HyTest platform. Therefore, the realization of different clients working collaboratively depends on their communication with the data center. To ensure successful data communication for HSMU, the data center should respond to the request of clients, as well as help synchronize the time step of multiple clients. At the beginning of a test, the clients assign a unique tag for each variable (i.e., displacements, forces, model parameters) to be sent to the data center, and request that the data center allocate storage space for these variables. Then the clients specify the variables to be read using the predefined tags. As for the time synchronization, the integration module determines the start and the end of a time step, which means all other clients should be synchronized with the integration module. For explicit time integration, the time synchronization is automatically achieved by all clients starting a new time step for every analysis loop. While for iterative implicit time integration, there are several trial analysis steps within a single time step, thus both time step and analysis step number should be transmitted together with data to provide the correct time information.

In this case, the data center is designed to store (1) the data mapping information for each client and (2) data value together with its unique tag, corresponding analysis step, and time step. TCP/IP communication protocol is adopted for remote access to the data center in the current implementation. Three major commands, ‘Initialize’, ‘Write’, and ‘Read’ are developed for clients to initialize space for storing data, write and read data, respectively. Each command is specified by a fixed data transmission format, as shown in Figure 3. Be aware that if a client is requesting for reading data which are unavailable right now but will be provided by other clients later, the data center will not respond to the request until these data are available. Typically for one analysis loop of a HSMU test, the test process for each client are listed as: the integration module first writes displacements to the data center, and requests to read restoring forces for solving the equation of motion; the experimental substructure reads displacements from the
data center first, and later writes measured restoring forces back; the identification module reads experimental displacements and restoring forces to identify material parameters, then writes these parameters to the data center; the numerical substructure reads displacements and identified parameters, and then writes numerical restoring forces back to the data center.

With the proposed data center-client architecture and uniform data transmission formats, multiple substructures as well as modules are enabled as independent clients. Each client initializes the information of data mapping, and guides its process of simulation or testing on its own. Hence, each client is treated equally from the data center's point of view. According to the data center, the difference between different substructures and modules is only reflected at the stage of initialization, when they specify different tags of data to be read and written. Due to this feature, both HSMU and conventional hybrid simulation can be conducted under the proposed architecture. In hybrid simulation, the numerical substructure only specifies the tags of displacements to be read, while tags of both displacements and parameters in HSMU.

3.2. Finite Element Identification Module

The identification module of HyTest aims to identify the material constitutive parameters of a finite element model according to experimental information online during the simulation. While there are several identification algorithms that can be used to determine the parameters of constitutive relationships, the unscented Kalman filter (UKF) [32] is chosen in this identification module. UKF is a powerful method in the field of system identification, and has been introduced into HSMU recently [10,13–17].

In a parameter estimation problem, the nonlinear discrete-time state-space model of a nonlinear system with Gaussian white noises can be described by the state transition equation

$$\mathbf{\theta}_k = \mathbf{\theta}_{k-1} + \mathbf{w}_k$$  \hspace{1cm} (2)

and the observation equation

$$\mathbf{y}_k = \mathbf{H}(\mathbf{\theta}_k, \mathbf{u}_k, \mathbf{p}_{k-1}) + \mathbf{v}_k$$  \hspace{1cm} (3)

where $$\mathbf{\theta}$$ is the vector of unknown parameters to be identified, and $$\mathbf{y}$$ is the measurement vector. $$\mathbf{w}, \mathbf{v}, \mathbf{u}$$ are the vectors of process noise, measurement noise, and model input, respectively. $$\mathbf{H}(\mathbf{\theta}_k, \mathbf{u}_k, \mathbf{p}_{k-1})$$ is the vector of observed outputs obtained from finite element model, in which $$\mathbf{p}_{k-1}$$ is the vector of historical variables that will be defined as follows. For a nonlinear structural model, the observation $$\mathbf{y}_k$$ is not only dependent on $$\mathbf{\theta}_k$$ and $$\mathbf{u}_k$$ as well as noise, but also on other variables $$\mathbf{p}_{k-1}$$ such as plastic strain for plastic constitutive model which reflects the effect of loading history. These variables are called as historical variables and denoted by $$\mathbf{p}$$ in this paper.

As an extension of Kalman filter (KF) to nonlinear systems, UKF utilizes a deterministic sampling technique called unscented transformation (UT) to help solve nonlinear problems without the assumption of linearization. The implementation of UT involves determination of $$2L + 1$$ (L is the number of parameters to be identified) parameter samples (sigma points), and calculation of corresponding measurements, which means $$2L + 1$$ computations with finite element model. It should be noted that historical variables are also part of the observation equation for nonlinear model. Therefore, all the $$2L + 1$$ computations should be based on the same historical variables, and once the UKF identified the current parameters, a $$2L + 2$$th computation is performed to generate correct historical variables for the next time step.

To implement the UKF identification procedure, OpenSees is used to simulate the observation function, and interact recursively with the identification algorithm programmed in MATLAB, as shown in Figure 4. The procedure herein is similar to that proposed by Astroza et al. [33], except for the differences on how to consider historical variables in finite element analysis to obtain observed outputs. In Astroza et al.’s approach, for every time new parameter samples were predicted, the finite element model was
re-analyzed from the beginning time $t_0$ to current time $t_k$ with new parameters. But in this paper, the analysis is performed for only one step from $t_{k-1}$ to $t_k$, by running OpenSees continuously with all historical variables saved in memory. Therefore, the identification process can be more time-efficient than Astroza et al’s method, and is suitable for online model updating.

Several modifications are made on the source code of OpenSees to accommodate this identification procedure. Due to the open-source and extensible framework of OpenSees, it is convenient to add new features. A class named ‘HyTestDataClient’ for external data communication is added in OpenSees, and another one named ‘StaticAnalysisForHyTest’ is added as a subclass of ‘StaticAnalysis’ to apply displacements and update constitutive parameters. In the original version of ‘StaticAnalysis’ as shown in Figure 5, a complete analysis step contains three basic operations, i.e., ‘newStep’ (to start a new step), ‘solveCurrentStep’ (to solve equations of current step), and ‘commit’ (to indicate the success of current time step and make current states as historical variables). While in the added ‘StaticAnalysisForHyTest’, firstly the ‘HyTestDataClient’ class is called to initialize a connection with MATLAB, and then receive data from Matlab. As the received data contains time step number as well as displacements and parameters, step information (with its value represented by ‘Initial TimeStep’, ‘Same TimeStep’, and ‘New TimeStep’) can be recognized. For a trial step which shares the ‘Same TimeStep’ with the previous step, ‘revertToLastStep’ will be triggered to retain the historical variables. While for a new time step, ‘commit’ command will be called to replace the historical parameters with current states. Then the functions ‘addSPConstraint’ and ‘updateParameter’ are performed to apply displacements, and update parameters, respectively. By performing ‘calculateNodalReactions’ instead of ‘commit’, restoring forces can be calculated, and finally ‘HyTestDataClient’ class is called again to send restoring forces back to MATLAB, and wait for data of the next step. It is worth noting that the class ‘HyTestDataClient’ is exactly the realization of functions described previously to access the data center of HyTest, and can be used to communicate with MATLAB as well. Hence, these modifications make OpenSees available not only to participate in parameter identification, but also to update the numerical substructure.

3.3. System Implementation and Interface Design

The data center of HyTest as a server has been developed based on C++, and the three major functions (‘Initialize’, ‘Write’, and ‘Read’) have been realized with various programming languages to help various clients access the data center. Currently these functions are available in C++ for OpenSees, Mex file based on C for MATLAB, LabVIEW for NI (National Instruments) system, and VB for MTS. Commonly there are three ways to use these functions with existing software. For software with extensible framework, these functions can be programmed as part of the software (i.e. ‘HyTestDataClient’ for OpenSees). While for others providing limited access to their framework (i.e., ABAQUS, MTS and NI system), wrappers can be used to implement these functions and interact with existing software. For example, by creating an external application which builds a file with structural model, and reads results from a database file, interaction with ABAQUS can be achieved [27]. The wrapper for ABAQUS has been realized using this method together with the C++ based functions for HyTest [30]. However this wrapper was developed primarily for conventional hybrid simulations. Although the wrapper could be employed for model updating, it is too time-consuming to be recommended as it requires re-analysis with the new parameters from the start for each step. The wrapper for NI system is implemented by LabVIEW graphical programming language, which will be described in the section 4.2. The last option is to directly compile these functions into the existing HS middleware OpenFresco to take advantage of the support provided for laboratory software.

To help simplify the operation of tests, a graphical user interface (GUI) named ‘Coordinator’ (Figure 6) including the integration module and data center is coded by C++, despite that the flow chart of integration module can also be implemented by other software (i.e., MATLAB). The user can easily use the system by following the GUI. Currently three basic integration algorithms, i.e., Central Difference, Explicit Newmark, and implicit Newmark-β, are available to be selected in the GUI. Another GUI named ‘MTS Connector’ has also been developed as the interface for MTS control system [34]. This interface is coded based on the VB COM provided by MTS, and can be used to send commands to MTS, acquire measured feedback, as well as communicate with the data center of HyTest.

4. Performance Validation of HyTest

In this section, two examples are presented to verify the proposed HyTest platform. In the first example, the identification module of HyTest is used to identify the material properties of a reinforced concrete column. Next, a steel frame is tested to evaluate the overall performance of HyTest as well as the ability of HSU to improve the fidelity of closed loop testing.

4.1. Material parameter identification of a reinforced concrete column

This example aims to evaluate the performance of the identification module of HyTest, with identifying constitutive parameters of concrete and steel together using the restoring force of a reinforced concrete column. The model designed for this validation is shown in Figure 7, defined with a nonlinear force-based element with five integration points along the element. The section is sub-divided into 144 fibers in the area of confined concrete, 32
fibers of unconfined concrete, and four steel bars. The constitutive model used for concrete is Kent-Scott-Park model [35], governed by the compressive strength \( f_c \), the compressive strain \( \varepsilon_0 \), the crushing strength \( f_d \), and the crushing strain \( \varepsilon_d \). The steel bars are described by bilinear model with three parameters, which are the yield strength \( f_y \), the Young’s modulus \( E \), and the strain hardening coefficient \( b_s \).

Prior to the identification phase, a sensitivity study on the restoring force has been conducted to select the material parameters to be identified. The \( f_c, \varepsilon_0, \) and \( f_d \) of confined concrete are selected for identifying concrete, as the restoring force of the column is not sensitive to \( \varepsilon_d \). And all three parameters \( f_y, E, b_s \) of steel will be identified. The initial estimates and true values of these parameters are shown in Table 2.

A lateral displacement history (Figure 8) is applied to the top node, which is equal to the dynamic response of the column subjected to El Centro (NS) excitation with true constitutive parameters. The measurement noise of the restoring force is assumed to be a zero-mean Gaussian white noise with 0.001 RMS noise-to-signal ratio. Finally the noisy restoring force history is sent to the previously described identification module step by step.

Two cases are studied to show the performance of proposed identification module, 1) identifying parameters of one material at a time (concrete and steel separately) with true constitutive parameters of the other material; 2) identifying parameters of concrete and steel together. Figure 9 shows the time histories of estimated parameters of concrete and steel.

For both cases, parameter \( f_d, f_y, \) and \( b_s \) remain equal to the initial values until about 1.5 sec because they are only related to the nonlinear properties. After the first peak valley of displacement command history at around 2.1 sec, all the parameters converge to their true values. Generally, in the case of identifying only one...
Table 2
Estimates and relative errors of identified parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>True value</th>
<th>Initial estimate</th>
<th>Final estimate by ID one material</th>
<th>Relative error</th>
<th>Final estimate by ID two materials</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_y) (MPa)</td>
<td>-31.80</td>
<td>-25.44</td>
<td>-31.81</td>
<td>0.03%</td>
<td>-31.65</td>
<td>0.47%</td>
</tr>
<tr>
<td>(\varepsilon_d)</td>
<td>-0.003855</td>
<td>-0.003084</td>
<td>-0.003865</td>
<td>0.26%</td>
<td>-0.003902</td>
<td>1.22%</td>
</tr>
<tr>
<td>(f_b) (MPa)</td>
<td>-23.85</td>
<td>-19.08</td>
<td>-23.85</td>
<td>0%</td>
<td>-24.35</td>
<td>2.10%</td>
</tr>
<tr>
<td>(E) (MPa)</td>
<td>375</td>
<td>300</td>
<td>375</td>
<td>0%</td>
<td>375.17</td>
<td>0.05%</td>
</tr>
<tr>
<td>(b_c)</td>
<td>210000</td>
<td>200000</td>
<td>210012</td>
<td>0.06%</td>
<td>209452</td>
<td>0.26%</td>
</tr>
<tr>
<td>(c)</td>
<td>0.030</td>
<td>0.024</td>
<td>0.030</td>
<td>0%</td>
<td>0.0298</td>
<td>0.67%</td>
</tr>
</tbody>
</table>

Fig. 6. Snap shots of proposed GUI ‘Coordinator’.

Fig. 7. Geometric and cross-sectional properties of a reinforced concrete column.

Fig. 8. Displacement history applied to the top node of column.

material, parameters converge faster than the case of identifying both concrete and steel.

Table 2 shows the relative errors of the final estimates based on true values. The maximum error occurs with the estimate of \(f_y\) in the case of identifying two materials together, which is 2.10%, and error of \(\varepsilon_d\) is 1.22%, while the errors of other parameters are less than 1%. Results indicate the effectiveness and accuracy of both identification cases, while the performance of identifying one material is better than two materials together. Moreover, the identification process is very efficient with the time consumption for both cases no more than 10 seconds for a total of 2000 steps.

4.2. HSMU of a multi-story steel frame

A series of hybrid tests were performed on a five-story steel moment frame using the HyTest platform. All stories are identical with the same geometric dimensions and material properties, and only the first story was included in the experimental substructure, whose dimension is shown in Figure 10. Both conventional hybrid simulation and HSMU were conducted at Intelligent Infrastructure System Laboratory, Purdue University. In the model updating approach, constrained UKF (CUKF) method was used to identify the material parameters. In this section, we focus on the test configuration for using HyTest and how HyTest meets the commands to conduct HSMU.

The testing system for HSMU is shown in Figure 11. Four parts of the system, which are the integration module, the numerical substructure, the experimental substructure, and the identification module, work cooperatively with HyTest handling all the data communication. As these four modules are separated with each other, the presented configuration is also suitable for hybrid simulation by removing the identification module.

The integration module, adopted the Central Difference method to solve the structural equation of motion. To simplify the dynamic model for illustration, the example structure is treated as a planar shear-type frame subjected to lateral excitation in one direction. As the basic frequency of the entire structure is about 4.14 Hz, the interval is set to 0.005 sec to ensure stability and accuracy. In this test, the C++ based GUI ‘Coordinator’ is used.

The numerical substructure, including the upper four stories, is modeled with OpenSees. Beams are assumed to be rigid without translational and rotational deformation. Columns are modeled using nonlinear force-based beam-column elements, with five inte-
Young's modulus $E$, and the strain hardening coefficient $b_s$ are updated at every step during the test.

The experimental substructure, selected as the first story, is loaded by one lateral actuator. The actuator is driven by voltage commands from Shore Western 6000 with inherent PID control, which is connected with a NI controller. The LabVIEW graphical programming environment is used to control NI hardware as well as acquiring data from both HyTest and DAQ cards. The LabVIEW interface contains two applications, one running in the RT target, which controls the actuator and gets measurement data, and another in the host computer, which accesses the data center of HyTest via TCP/IP. The two applications exchange displacement commands and force feedbacks through networked-published shared variables [36], as shown in Figure 12. In the LabVIEW application running in the host computer, three functions ('Initialize', 'Read', 'Write') provided by HyTest are called to initialize space for storing experimental data, as well as read and write data. The displacement command received from HyTest is interpolated to five points to guarantee smooth loading, with four points at the ramp stage while the last one at the steady stage. The time delay between 'Read' and 'Write' is set to 750 ms to ensure enough loading time. While in the application running in RT target, the loop starts by reading command displacement from a shared variable, then loading, and at last storing measurement force as a shared variable. Each loop is running at a period of 100 ms to match the time delay setting and the interpolation strategy in the host application, which means seven or eight real-time loading loops are performed for every integration time step. In this case, LabVIEW bridges the data center of HyTest and experimental hardware.

The identification module, as described earlier in this paper, is used to identify the material properties of the experimental substructure. A corresponding numerical model of the experimental substructure with undetermined material parameters is built in OpenSees, and the CUKF algorithm is coded within the identification module to identify these undetermined parameters. Like the configuration of LabVIEW application, the MATLAB program contains two parts, one for assessing data in the data center of HyTest, and the other for parameter identification. As both MATLAB and OpenSees are running on non-real-time Windows, these two parts are coded in a single MATLAB application, by starting the loop with reading displacement and force from the data center of HyTest, then identifying parameters together with OpenSees, and sending these parameters to the data center of HyTest at last.

In order to evaluate the impact of HyTest in a hybrid simulation, the original entire structure was first analyzed with OpenSees through a numerical simulation. Then HyTest was used to conduct a purely numerical hybrid simulation (virtual HS) in a substructured fashion, with the two substructures modeled by OpenSees separately. Finally the real hybrid test using the test configuration above was performed with HyTest. In these three cases, the material parameters used for numerical analysis were the initial estimates as listed in Table 3. Later, a hybrid test with online model updating was performed to validate the modified OpenSees for updating parameters, as well as the overall performance of HyTest.
As the identification module had already been verified previously, the converged values of material parameters (Table 3) obtained by HSMU were assumed to be the exact ones. These parameter values were then used to analyze the entire structure as the reference solution. Table 4 summarizes the description for all these five cases.

Figure 13 compares the hysteresis curves and displacement histories of the first story with numerical simulation, virtual HS, and HS. As can be seen in both Figures 13 a) and b), the use of HyTest barely has little influence on the numerical results. The quite slightly difference was caused by the simplification of structural dynamic model. In the virtual hybrid simulation, the entire

---

**Table 3**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial estimate</th>
<th>Final estimate by HSMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_y$ (MPa)</td>
<td>600</td>
<td>352.73</td>
</tr>
<tr>
<td>$E$ (MPa)</td>
<td>200000</td>
<td>206000</td>
</tr>
<tr>
<td>$b_1$</td>
<td>0.045</td>
<td>0.1464</td>
</tr>
</tbody>
</table>

---

**Fig. 11.** HSMU testing system of a five-story steel frame using HyTest.

**Fig. 12.** Communication strategy between non-real-time host application and real-time application by LabVIEW.

**Fig. 13.** Comparison of the first story response between numerical (with and without HyTest) and experimental results.
Table 4
Simulation and testing cases.

<table>
<thead>
<tr>
<th>Cases</th>
<th>Experimental Substructure</th>
<th>Numerical Substructure</th>
<th>Model Updating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerical Simulation</td>
<td>Entire structure modeled with initial estimates</td>
<td>Modeled with initial estimates</td>
<td>NA</td>
</tr>
<tr>
<td>Virtual HS (HyTest)</td>
<td>Modeled with initial estimates</td>
<td>Modeled with initial estimates</td>
<td>NA</td>
</tr>
<tr>
<td>HS (HyTest)</td>
<td>Loaded experimentally</td>
<td>Modeled with initial estimates</td>
<td>NA</td>
</tr>
<tr>
<td>HSMU (HyTest)</td>
<td>Loaded experimentally</td>
<td>Modeled with undetermined parameters in OpenSees</td>
<td>CUKF identification</td>
</tr>
<tr>
<td>Reference</td>
<td>Entire structure modeled with final estimates</td>
<td></td>
<td>NA</td>
</tr>
</tbody>
</table>

Fig. 14. RMSD of story displacements and restoring forces.

structure was approximated as a shear-type model with a lumped mass at each story, hence mass and stiffness matrices were both reduced to five by five elements, while the OpenSees model had a higher density of meshing. However, the hybrid simulation result at this level of mesh refinement is still acceptable. Furthermore, the hysteresis behavior of the first story shows a nearly perfect agreement between the virtual hybrid simulation and real hybrid test, which indicates the successful strategy of proposed HSMU configuration.

The root-mean-square deviation (RMSD) of story displacements and restoring forces were calculated to describe the difference between HSMU and reference results, as well as that between HS and reference. With updating the material parameters of the numerical substructure, which is the second to fifth story, HSMU shows better accuracy compared to HS on the both displacements and restoring forces (Figure 14). Of all the stories, the displacement response of the second story has been improved most significantly. Hence, a deeper look is taken at the hysteresis curves and displacement histories of the second story as shown in Figure 15. The nearly perfect match between HSMU and reference results validates that the material parameters in OpenSees model has been successfully updated, and the nonlinear behavior was captured using HSMU.

The HSMU test required about 165 minutes for a 60 seconds El Centro ground motion, of which nearly 125 minutes were allocated to experimental control and loading. The time consumed by the integration module, numerical simulation, and identification were quite small as depicted in Table 5. As all the unphysical modules were running on the same computer, the data transmission time was negligible. The low cost on computational time indicates the possibility to run a fast hybrid simulation.

5. Conclusion

The software platform HyTest is extended to support parameter updating of numerical substructures modeled in sophisticated finite element software such as OpenSees during hybrid simulations. An identification module is developed to implement the UKF method that has been intensively investigated in recent years. As the parameter sampling in UKF identification involves repeated evaluations of the auxiliary finite element model for the experimental substructure, the computational cost may be huge with the original OpenSees or other finite element software. To this end, the
source code of OpenSees is modified to provide a time-efficient procedure to conduct these repeated evaluations. The evaluation of the auxiliary finite element model with each set of sampled constitutive parameters is performed from the same initial state at the start of current step, which is the final state of the auxiliary model with the parameters updated at the end of previous time step. The so-developed new version of HyTest is validated by the numerical analysis on parameter identification of a reinforced concrete column and the hybrid simulation on a steel frame with model updating. Although the validating examples are only concerned with parameter updating of constitutive laws on material level, HyTest can also be applied for the structures with more complex nonlinear behaviors if only the nonlinearities can be represented by the constitutive models on any level (material, sectional, or element level) within the framework of OpenSees.

However, it should be noted that the model updating method employed in this paper may encounter challenges when the structures experience abrupt change of restoring forces, such as crushing or rupture of reinforced concrete and steel structures under extreme earthquakes. It is certainly an important and interesting topic worthy of further studies.

Acknowledgments

This research is financially supported by the National Key Research and Development Program of China (Grant No. 2016YFC0701106), the National Natural Science Foundation of China (Grants No. 5116120360 and 9135301-09), and the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20132302110065). The third author would like to also acknowledge the Purdue University Bilandz vice Fellowship for Financial support. Experiments were conducted using a NSF-supported facility developed within the Intelligent Infrastructure Systems Lab at Purdue.

References