# Cyber-Physical Systems for Real-Time Hybrid Structural Testing: A Case Study *

Huang-Ming Huang, Terry Tidwell, Christopher Gill, Chenyang Lu
Dept. of Computer Science and Engineering, Washington University, St. Louis, MO, USA
{hh1, ttidwell, cdgill, lu}@cse.wustl.edu

Xiuyu Gao[1], Shirley Dyke[2,1]
[1]School of Civil Engineering, Purdue University, West Lafayette, IN, USA
[2]School of Mechanical Engineering, Purdue University, West Lafayette, IN, USA
{xygao, sdyke}@purdue.edu

## ABSTRACT

*Real-time hybrid testing of civil structures, in which computational models and physical components must be integrated with high fidelity at run-time, represents a grand challenge in the emerging area of cyber-physical systems. Actuator dynamics, complex interactions among computers and physical components, and computation and communication delays all must be managed carefully to achieve accurate tests.*

*In this paper we present a case study of several fundamental interlocking challenges in developing and evaluating cyber-physical systems for real-time hybrid structural testing: (1) how physical and simulated components can be integrated flexibly and efficiently within a common reusable middleware architecture; (2) how predictable timing can be achieved atop commonly available hardware and software platforms; and (3) how physical vs. simulated versions of different components within a system can be interchanged with high fidelity between comparable configurations. Experimental results obtained through this case study give evidence of the feasibility and efficacy of these steps towards our overall goal: to develop a* Cyber-physical Instrument for Real-time hybrid Structural Testing (CIRST).

## Categories and Subject Descriptors

C.3 [**Special-purpose and Application-based Systems**]: Real-time and Embedded Systems

## General Terms

Performance, Experimentation

*This research was supported in part by NSF grants CNS-0821713 (MRI), CNS-0448554 (CAREER), and CCF-0448562 (CAREER).


*ICCPS'10*, April 13–15, 2010, Stockholm, Sweden.


## Keywords

Real-time hybrid structural testing, cyber-physical systems, middleware

## 1. INTRODUCTION

Structural Engineering increasingly relies on sophisticated computational monitoring and control systems involving mechanical and structural components, and these cyber-physical systems must be tested and validated using similarly sophisticated techniques. While high-fidelity validation is critical to the acceptance of structural monitoring and control systems, in many applications testing numerous possible scenarios of a new structural control device or a new monitoring system is not feasible due to the cost and time required for such a comprehensive test. For instance, in developing civil infrastructure, the performance of a new vibration suppression system (e.g., for earthquake or hurricane mitigation) usually cannot be validated at full scale prior to its implementation (e.g., on a large bridge). In the current state of the art, the cost of such testing prohibits performing more than a few representative tests at small scales relative to the massive sizes of the structures, and thus additional validation steps are needed. Furthermore, these tests are often destructive, so that only one test can be performed with each test specimen.

While reduced scale testing is useful, it cannot always capture important behaviors of the full scale structure, even if scaling effects have been carefully considered. Numerical simulation is therefore an equally important technique in modern structural analysis, and has benefited significantly by leveraging hardware innovations that offer improved computational capabilities. However, experimental validation is still essential to examine the underlying assumptions made by the numerical models, especially considering the existence of highly nonlinear elements under extreme dynamic loading.

*Hybrid testing*, which integrates physical components of a structure of interest with computational models of other known structural components, thus improves significantly on either purely numerical or purely empirical approaches. Due to a lack of *real-time* hybrid testing support, however, hybrid testing at a slow (*pseudodynamic*) time scale is the state

of the art [10, 11, 17, 16]. Unfortunately, testing at such time scales may not reveal critical dynamic system features, which motivates a real-time approach. The leap to real-time raises significant research challenges such as meeting stringent requirements on timing and testing fidelity, which motivate the development and use of new cyber-physical infrastructure for real-time hybrid testing.

In prior work [22] we developed an initial prototype of a *Cyber-physical Instrument for Real-time hybrid Structural Testing* (CIRST), to illustrate our vision and gain insights into key design and implementation challenges. However, general applicability of that prototype is restricted by the following limitations, which the work presented in this paper addresses: (1) the prototype is difficult to extend or modify since it was implemented natively rather than atop reusable and reconfigurable middleware; and (2) although timing behavior is fundamental to establishing a general cyber-physical capability for real-time hybrid structural testing, timing issues were not studied explicitly in [22].

To overcome those limitations this paper presents three new contributions to the state of the art in cyber-physical systems for real-time hybrid structural testing: (1) a reusable middleware architecture and its efficient implementation in C++, within which both cyber and physical components can be integrated flexibly through XML-based configuration specifications; (2) a case study of real-time hybrid structural testing scenarios involving reconfigurable integration and substitution of cyber and physical components using that middleware; and (3) what is to our knowledge the first measurement-based evaluation of real-time hybrid structural testing atop reconfigurable middleware technology. The results of our evaluation serve to quantify timing behavior, demonstrate the feasibility and efficacy of our middleware-based approach, and reveal new insights into real-time hybrid structural testing such as the potential of systems to tolerate small numbers of timing constraint violations, and how interactions between cyber and physical components relate to overall system behavior.

Section 2 describes requirements for real-time hybrid testing, which motivate and guide our approach. Section 3 surveys related work. Section 4 discusses the design and implementation of the CIRST middleware, and Sections 5 and 6 present our case study and experimental evaluations using it. We summarize the contributions of this research and describe future work in Section 7.

## 2. MIDDLEWARE REQUIREMENTS

In this section we describe requirements for real-time hybrid structural testing, which fall into three main categories: *encapsulation*, *specialized infrastructure*, and *flexible configuration*. A crucial feature of these requirements is their inherently cyber-physical nature, which differentiates them from traditional real-time and embedded systems requirements. For example, (1) encapsulation includes the need for unified abstractions that are relevant to both cyber and physical elements of a real-time hybrid structural testing system; (2) specialized infrastructure is needed to ensure functional and temporal correctness and optimize performance of data and event flows not only within but *between* cyber and physical portions of a system; and (3) declarative deployment and assembly *of those unified abstractions and the supporting infrastructure* is needed to help structural engineers (who may have limited familiarity with programming languages like C++ but are expert with natural abstractions in the structural testing domain) construct, execute, and reconfigure experiments.

***Encapsulation.*** The first requirement we consider is that system middleware must provide suitable mechanisms to encapsulate both cyber and physical abstractions under a unified model, so that different components of a real-time hybrid structural testing system can be added, removed, or interchanged without inordinate impact on other system components. For example, a small scale experiment may use a physical actuator and a physical test specimen to obtain models of their behaviors, and then a larger scale experiment may integrate different simulated components (based on those models) with the physical test specimen and actuator. This kind of encapsulation is the norm in existing hybrid testing platforms such as dSPACE [1] (which leverages Simulink's approach to building systems from constituent *blocks*) and OpenSees [5] (which offers an object oriented approach to specifying experiments, thus promoting reuse and addressing design complexity). Any cyber-physical middleware for *real-time* hybrid testing must offer comparable encapsulation capabilities.

***Specialized Infrastructure.*** While the previously mentioned requirement has led to the design of reusable middleware frameworks (e.g., CIAO [20], TAO [15], nORB [21] and ACE [18]) to constrain system complexity while maintaining real-time performance for other application domains, the nuances of real-time hybrid testing require further innovation in middleware design and implementation beyond those frameworks. The scale and complexity of experiments that already can be conducted using existing hybrid testing approaches such as dSPACE [1] and OpenSees [5] impose stringent performance requirements for middleware that would support similarly rich hybrid testing in *real-time*. To achieve suitable real-time performance, data and event flows through the system must be flexible, correct, efficient, and temporally predictable even at small time scales (e.g., for hybrid testing of a multi-story civil structure at 1280Hz).

***Flexible Configuration.*** The third requirement imposed by real-time hybrid testing is that configuration capabilities must allow system developers to specify *both declaratively and flexibly* which elements will constitute the system, and how those elements will be initialized and interconnected. Such configurability allows both recurring and new hybrid structural tests to be assembled and deployed readily from common sets of reusable components.

In addition to easing the task of specifying system configurations, these capabilities also must reduce the risk of configuration errors. Our previous experience developing a hand-coded prototype for simple hybrid testing experiments [22] often involved tedious and error-prone experimental configuration changes, which motivates the importance of this requirement. Without suitable tools for system wide manage-

ment of cross-cutting cyber-physical concerns, implementing those experimental configuration changes required code changes throughout the prototype system, which in turn risked introducing further problems that would need to be addressed.

# 3. RELATED WORK

*Target Systems.* Control systems for hybrid testing can be specified using high level Simulink models, compiled into object code, and then linked with a light weight real-time kernel which provides basic interrupt and I/O services to generate executable code. dSPACE's TargetLink [1] provides a Simulink model compiler and a real-time kernel to produce executables that can be downloaded and run on specialized hardware platforms. The xPC target [8] from Mathworks is a similar toolset that targets generic x86 hardware instead. For example, the platform developed at UIUC [12] targets more generic x86 hardware, allowing greater flexibility in the experimental equipment. The major benefit of this approach is to support model definition, evaluation, and target deployment without requiring significant programming for basic scenarios. For more complicated hybrid testing systems with hundreds of degrees-of-freedom, more complex nonlinear material and structural models are needed, which are core elements of the OpenSees [5] open source structural analysis framework adopted by NEES [19]. OpenSees also uses object oriented programming to provide tools for specifying reusable numerical models for simulations. OpenSees is purely for computation of the response of the numerical model, and depends on a real-time OS like ETS [2] which NEES uses for real-time operation. As we noted in [22], specialized software and hardware can easily run into the tens of thousands of dollars, whereas our goal with CIRST is to support high-fidelity real-time hybrid structural testing at lower cost and with greater flexibility, using off-the-shelf platforms such as Linux.

*Middleware and Tools.* Real-time middleware like nORB [13], TAO [15], and CIAO [14] all assume a distributed *object* computing model. The CIRST middleware instead provides data-flow oriented component abstractions within a lightweight implementation. The MARTE[4] UML profile allows specification tools from different vendors to interoperate. Real-time and embedded systems environments such as Ptolemy [6] and Metropolis [3] support formal modeling, design, and analysis of cyber-physical system properties. Our research is currently focused on fundamental middleware issues, though code generation and verification from detailed formal models may be beneficial as future work.

# 4. MIDDLEWARE APPROACH

To address the requirements discussed in Section 2, the CIRST middleware is designed to: (1) *encapsulate* system abstractions as components, (2) provide *specialized infrastructure* to enforce type safety and timing properties within and between components, and (3) provide *flexible configuration* capabilities end-to-end. Our approach leverages well-known middleware features like components, ports, local and remote invocation, and XML-based configuration, but then adapts and extends them in novel ways to address the nuances of real-time hybrid structural testing. The CIRST middleware is the foundation for what is to our knowledge the first exploration and empirical evaluation of reusable middleware technology in real-time hybrid structural testing (presented in Sections 5 and 6), which is an important contribution of this paper.

## 4.1 Encapsulation

The CIRST middleware is designed to encapsulate both cyber and physical elements of a system as distinct *components*. There is no notion of objects or inheritance in the CIRST middleware, since real-time hybrid structural testing is largely *data-flow* (instead of *object*) *oriented*. Communication between components is based on a set of defined input and output interfaces, which we refer to as *flow ports* since different data types can be configured to flow through them from an *out-port* of an *upstream* component to an *in-port* of a *downstream* component. This allows lower-level details about how ports communicate (e.g., over a network vs. over local or shared memory) to be decoupled from the components themselves.

## 4.2 Specialized Infrastructure

To achieve sub-millisecond time scales atop common-off-the-shelf platforms like Linux, the CIRST middleware infrastructure is designed to achieve type safety and temporal predictability within a lightweight C++ implementation supporting local and remote interactions among components. The CIRST middleware infrastructure has three important areas of specialization: *component ports*, *data movement optimization*, and *timing constraint handling*.

Component ports enforce type safety by checking compatibility of the data type sent by an out-port and the data type expected by an in-port. These checks are done statically using C++ templates, and do not add run-time execution cost or jitter. The CIRST middleware allows the out-port of a component to be connected to multiple in-ports of other components and preserves the integrity of data passed between the connections such that all downstream components see the same values. If a test structure is sufficiently complex, the computation time (e.g., of a finite element method) may be longer than the period between inputs from a physical sensor or outputs to a physical actuator, in which case the numeric computation must be partitioned and dispatched to different physical processors for real-time parallel computation. The need for such flexible encapsulation illustrates how the design of middleware for *cyber-physical* systems is shaped by rich potential interactions *among* cyber and physical elements.

The CIRST middleware infrastructure can reduce data copying in some cases by allowing an upstream component to declare whether the data it sends can be modified by its downstream components. This in turn allows downstream components within the same process to decide whether to copy its input data or reuse the input buffer. Unlike the other specializations in this section, this data movement optimization only can be applied between components co-located within the same process address space. We emulate the r-value reference (or *move semantics*) in C++0x [9] but use a special `Moveable` template that allows us to work around a potential type safety problem with the current C++ standard's

`std::tr1::function` being unable to distinguish correctly between overloaded functions with const vs. non-const references.

Finally, the CIRST middleware allows timing constraints to be associated with a component and appropriate handlers to dispatched if a constraint is violated at run-time. These handlers can be configured with different policies: to stop the experiment, mitigate the violation and continue the experiment, or even ignore certain violations.

## 4.3 Flexible Configuration

To ease configuration via declarative specifications, and to make it accessible to users without a C++ background, the CIRST middleware provides a scripting capability to set up and run a cyber-physical experiment. An XML configuration file specifies the components used by an experiment, the connections between components, the configuration parameters and timing constraints for each component, and the timing instrumentation points to be used for performance analysis. We achieve this capability through a code generator that reads the XML configuration file and emits C++ code that can be compiled into an executable program. The configuration file also can be read by the executable program when it starts up, to change the component configuration parameters without regenerating or re-compiling C++ code.

If components cannot be connected directly due to data type mismatches, but an acceptable data conversion exists (e.g., between different units of measure) the CIRST middleware allows users to specify data conversion expressions that are parsed by the code generator and used to generate C++ components that perform the specified conversion. By relieving system developers of the tedious and error-prone responsibility either to ensure exact type matching among component ports or to code C++ components to perform the necessary conversions, the automatic generation of these adapters further assists system developers in using the CIRST middleware.

# 5. CASE STUDY

In this section we present a case study that describes a representative experimental setup for real-time hybrid testing and shows how the CIRST middleware can be applied to develop and run different cyber-physical experiments. In Section 6 we present experimental results obtained in this case study, which confirm the efficacy of the CIRST middleware for (1) running accurate real-time hybrid tests, (2) comparing the performance of alternate configurations of physical and computational components in those tests, and (3) yielding valuable insights into the behavior of cyber-physical experiments run atop it.

## 5.1 Experimental Setup

The *physical test specimen* in our experimental setup is composed of a small steel compression spring, which is used to represent the bending stiffness of an actual column in a portal frame structure. The linear elastic spring has a nominal stiffness of 37.6 kN/m (215 lb/in) and a maximum allowable deflection of 7 cm (2.77 in). The rest of each structural frame is modeled in a *computational substructure* that assumes: (1) each column has the same stiffness as the physical test specimen, and (2) classical proportional damping of 2% for each structure mode. LA16 earthquake acceleration data [7] taken from the Woodward-Clyde Federal Services database (shown in Figure 1) are used as a ground motion input to excite the computational substructure. The data are scaled down in magnitude to avoid having the maximum structure response exceed the deflection limit of the experimental setup.

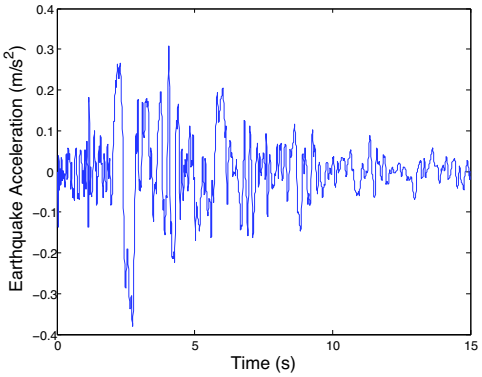



Figure 1: Input ground motion

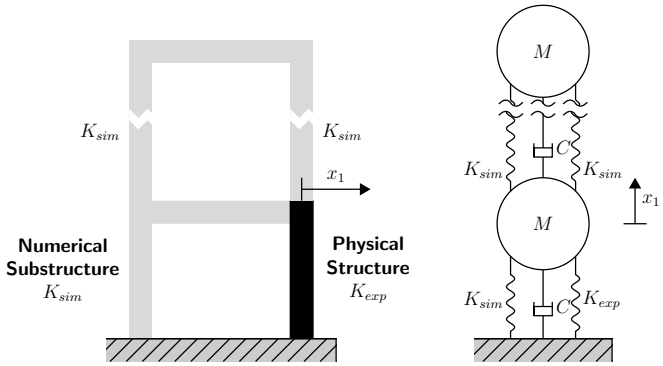



Figure 2: Experimental hydraulic structure testing model

A Shore-Western 910D double-ended hydraulic *actuator* is employed as the loading device to drive the physical test specimen. The actuator has a maximum stroke of 6 inches, with a built-in concentric linear variable differential transformer (LVDT) for ready integration into a position feedback control system. A Schenck-Pegasus 162M servovalve rated for 15 GPM at a 1,000 psi pressure drop is used to control the actuator. The servo-valve has a nominal operational frequency range of 0-60 Hz and is driven by a Schenck-Pegasus 5910 digital controller. An Omega load cell with a range of 1 kip is included in series with the physical test specimen to measure the restoring force.

The experimental model and setup are shown in Figure 2 and Figure 3 respectively. Figure 4 illustrates two alternative specializations of the experimental model, which we used to test substitution of physical and simulated components: (a) with the physical structure on the bottom floor, and (b) with the physical structure on the top floor of a two-

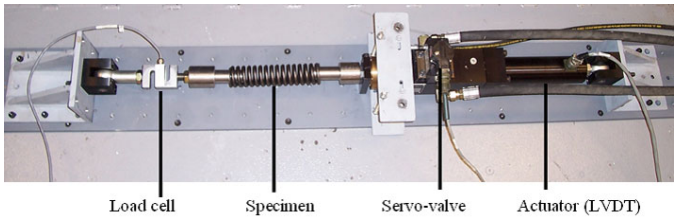
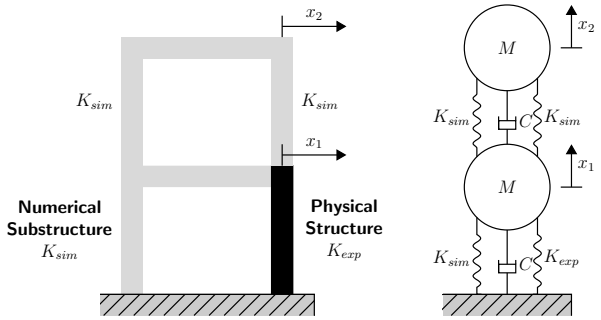
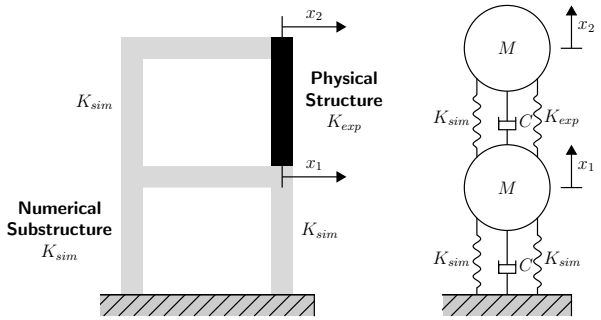



Figure 3: Experimental setup for hydraulic structure testing




(a) Physical structure on bottom floor




(b) Physical structure on top floor

Figure 4: Model specializations for component substitution

story structure. Experiments conducted using this setup and these models were performed in the Washington University Structural Control and Earthquake Engineering Laboratory, using a hydraulic pump that can be operated at 3,000 psi with maximum flow rate of 43 GPM.

## 5.2 System Control Model

An experimental transfer function for the overall physical component was obtained under a band-limited white noise excitation signal with a bandwidth of 50 Hz and magnitude root mean square of 0.07 cm (0.028 inch). Actuator parameters were identified using a nonlinear least âĂŞsquare optimization routine to curve-fit the experimental data, and the value of each parameter as well as its physical meaning is shown in Table 1.

As can be seen in Figure 5, the curve-fitted model represents the actuator dynamics well within a 0-50 Hz frequency range. For this reason we used it in our controller design to compensate for actuator dynamics. Since stability and accuracy are the major concerns for dynamic analysis, the applied hybrid testing methodology is studied by comparing a prototype single degree-of-freedom (DOF) hybrid test sys-

| | | | |
|---|---|---|---|
| $K_p$ | 3 | $mA/in$ | controller proportional gain |
| $\tau_v$ | 2.76e-3 | $s$ | servo-valve time constant |
| $K_q$ | 26.959 | $in^3/s/mA$ | valve flow gain |
| $K_c'$ | 2.499e-4 | $in^3/s/psi$ | valve flow pressure gain |
| $A$ | 0.652 | $in^2$ | piston area |
| $C_l$ | 2.476e-5 | $in^3/s/psi$ | piston leakage coefficient |
| $V_t$ | 36.59 | $in^3$ | volume of fluid |
| $\beta_e$ | 96153 | $psi$ | effective bulk modulus |
| $m_t$ | 1.754e-2 | $lb-s^2/in$ | mass of test specimen |
| $c_t$ | 8.429 | $lb-s/in$ | viscous damping coefficient |
| $k$ | 212.62 | $lb/in$ | stiffness of specimen |

Table 1: Identified actuator parameters

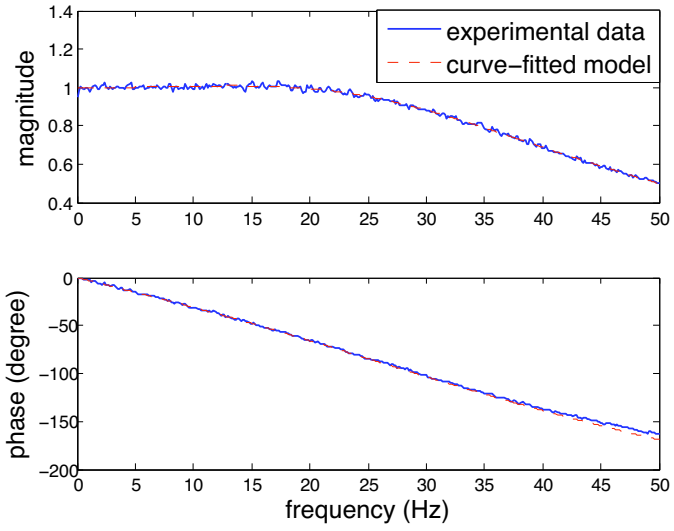



Figure 5: Experimental transfer function vs. model

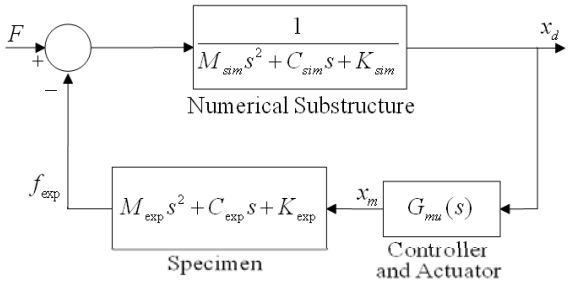



Figure 6: Closed-loop hybrid test system

tem with a reference system whose input (excitation force) and output (displacement) relationship is governed by the transfer function in Equation 1:

$$G_{anyl} = \frac{1}{Ms^2 + Cs + K} \tag{1}$$

The closed-loop hybrid test system is shown schematically in Figure 6 and can be described by the transfer function shown in Equation 2. Equation 3 represents the experimental components' transfer functions between the input command and output displacement measurements.

$$G_{hyd} = \frac{1}{\begin{array}{l} M_{sim}s^2 + C_{sim}s + K_{sim} + \\ G_{mu}(s) \cdot [M_{exp}s^2 + C_{exp}s + K_{exp}] \end{array}} \tag{2}$$

$$G_{mu}(s) = \frac{K_p \frac{K_q A}{K_c}}{H(s)} \tag{3}$$

where $H(s) =$

$$(\tfrac{V_t}{4\beta_e K_c} m_t \tau_v)s^4 + (\tfrac{V_t}{4\beta_e K_c} m_t + m_t \tau_v + \tfrac{V_t}{4\beta_e K_c} c_t \tau_v)s^3 +$$

$$(m_t + \tfrac{V_t}{4\beta_e K_c} c_t + \tfrac{A^2}{K_c}\tau_v + c_t \tau_v + \tfrac{V_t}{4\beta_e K_c} k \tau_v)s^2 +$$

$$(c_t + \tfrac{V_t}{4\beta_e K_c} k + \tfrac{A^2}{K_c} + k\tau_v)s + k + K_p \tfrac{K_q A}{K_c}.$$

Two complex conjugate poles of the hybrid test system represent structure poles that are modified in the complex plane due to the servo-hydraulic system. These poles dominate the dynamic characteristics of the hybrid test system, as opposed to four other poles with much faster structural dynamics. Assuming that the total mass, total damping and half of the stiffness are modeled in the computational substructure, Tables 2 and 3 compare natural frequencies and damping ratios respectively, between the reference system (Sys-1) and the hybrid test system with (Sys-2) and without (Sys-3) compensation.

Although the natural frequency variation is small, significant damping reduction has been observed in the hybrid test system. This can be improved by the use of the compensation scheme. More damping reduction is associated with the test when the natural frequency of the reference system increases, which indicates a larger error introduced by the hybrid testing methodology. Instability will occur if the damping becomes negative.

| | Natural Frequency (Hz) | | | |
|---|---|---|---|---|
| Sys-1 | 1 | 2 | 5 | 10 |
| Sys-2 | 1 | 2 | 5.0001 | 10.002 |
| Sys-3 | .9999 | 1.9996 | 4.993 | 9.9469 |

Table 2: Natural frequencies: $K_{exp} = K_{sim} = K$

| | Damping | | | |
|---|---|---|---|---|
| Sys-1 | 0.02 | 0.02 | 0.02 | 0.02 |
| Sys-2 | 0.0177 | 0.0153 | 0.0088 | 0.0007 |
| Sys-3 | 0.0083 | -0.0034 | -0.0374 | -0.0851 |

Table 3: Damping ratios: $K_{exp} = K_{sim} = K$

Consider also another (worst) case with zero stiffness but again with the total mass and total damping modeled in the computational substructure. Tables 4 and 5 show natural frequencies and damping ratios for this test scenario, with faster damping reduction indicating degraded stability and accuracy. Through this simple example, we can conclude that overall hybrid test system behavior depends not only on local behaviors such as accurate modeling and a good compensation strategy, but also intrinsically on the experimental plan and setup, e.g. on the proper partition of physical properties between numerical and experimental elements of the system.

## 5.3 Middleware Configuration

To examine how the CIRST middleware can be applied in different cyber-physical experiments using the experimental setup described in Section 5.1 and the system control model described in Section 5.2, we configured a component assembly that integrates a commercially available I/O device and C++ and Matlab simulation components on a standard Linux platform.

The CIRST middleware component assembly, implemented by C++ code generated from an XML configuration file, is shown in Figure 7. We ran the component assembly on an Intel® Core™2 Quad 2.66 Hz CPU machine with four gigabytes of RAM. For communication with the analog sensors and actuators in the experimental setup, we used a National Instruments Data Acquisition Board (BNC-2120 DAB). The `DaqReader` and `DaqWriter` components in Figure 7 represent the software components for reading from and writing to the data acquisition board.

| | Natural Frequency (Hz) | | | |
|---|---|---|---|---|
| Sys-1 | 1 | 2 | 5 | 10 |
| Sys-2 | 1 | 2 | 5.0003 | 10.006 |
| Sys-3 | .9999 | 1.9991 | 4.9875 | 9.9534 |

Table 4: Natural frequencies: $K_{exp} = K$, $K_{sim} = 0$

| | Damping | | | |
|---|---|---|---|---|
| Sys-1 | 0.02 | 0.02 | 0.02 | 0.02 |
| Sys-2 | 0.0153 | 0.0107 | -0.0024 | -0.0185 |
| Sys-3 | -0.0035 | -0.0268 | -0.0947 | -0.1898 |

Table 5: Damping ratios: $K_{exp} = K$, $K_{sim} = 0$

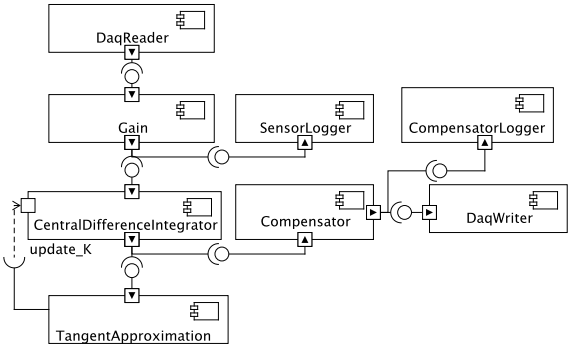



Figure 7: Component assembly

Each simulation step triggers a cascade of execution following the order of flow ports in the assembly, which generates commands to the actuators. We first used the data acquisition board's hardware clock to trigger each simulation step. However, unforeseen timing variation with that mechanism led us to replace it with a more regular software based triggering mechanism, an experience that further illustrates the utility of a configurable component based approach to real-time hybrid structural testing.

# 6. EXPERIMENTAL EVALUATION

To evaluate the scalability of our middleware, we first extended our experiment to use a sequence of $n$-story structures (for different values of $n$) each of which is equivalent in our system model to an $n$ degree-of-freedom (DOF) system; however, only one column of the bottom floor is captured from the physical rigid beam specimen, and data for all other stories are purely based on analytical computation. We used a 15 second earthquake record and a simulation frequency of 1280Hz. For each value of $n$ we ran 10 trials. To assess the system's ability to perform at the specified simulation frequency we recorded the number of times that a deadline was missed (i.e, when a component's execution completed after the end of the current period at that simulation frequency). The results of these trials, sorted by the number of deadline misses, are shown in Figure 8. As those results show, the real-time hybrid test system experienced some (though very few) deadline misses as $n$ goes up to 255. After that, however, the computation load becomes too large for the system to maintain at 1280 Hz.

## 6.1 Attribution of Deadline Misses

To examine the sources of these deadline misses, we instrumented the system to measure the time required for the following operations: reading sensor data, writing actuator commands and performing other numerical computations. Figure 9 shows the cumulative distributions for each of the

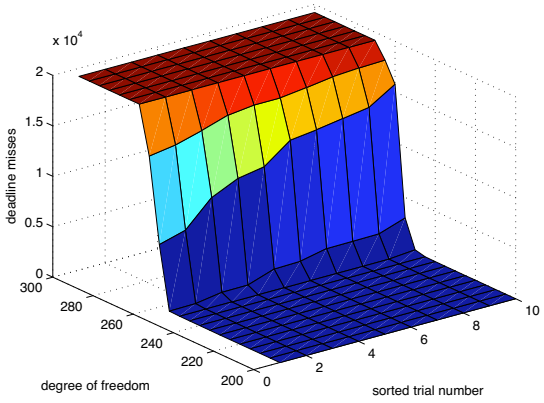



Figure 8: Deadline misses for $n$-story structures

| | 5 | 200 | 280 |
|---|---|---|---|
| read | 177.74 $\mu$sec | 170.13 $\mu$sec | 186.7 $\mu$sec |
| write | 182.43 $\mu$sec | 157.84 $\mu$sec | 210.8 $\mu$sec |
| computation | 131.79 $\mu$sec | 645.82 $\mu$sec | 1140.2 $\mu$sec |
| total | 272.38 $\mu$sec | 844.66 $\mu$sec | 1433.1 $\mu$sec |

Table 6: Worst case time for each simulation step segment

operations individually and in aggregate, when $n$ is 5, 200 and 280. Those distributions indicate that reading from and writing to the DAQ board dominated the aggregate timing when $n$ is 5, but as the number of degrees of freedom increased the relative contribution of the other (computational) components of the system made up more and more of the aggregate timing. Despite this shift in relative contribution, the DAQ read and write operations exhibited more and more variability in timing as the number of degrees of freedom increased. These results demonstrate that I/O timing, I/O timing variability, or both may differ across different real-time hybrid testing experiments, and that making similar measurements for each configuration appears valuable.

Table 6 shows the worst case scenario for those segments. In the case of $n = 200$, the average total execution time is 491.69 $\mu$s , which is well within our deadline of 780 $\mu$s. However, the worst case is 844.66 $\mu$s, which happens when the other computation operation spends 646.73 $\mu$s.

## 6.2 Fidelity Comparisons

We compared the fidelity of our hybrid test results obtained in the presence of those deadline misses, to comparable tests using pure simulation models that did not suffer that effect. Figures 10, 11 and 12 show the results of hybrid vs. simulation-only tests with 1, 5, and 60 degrees-of-freedom respectively. Each DOF indicates one lumped mass. The single DOF test assumes a 1910 kg floor mass and the other two tests assume a 175 kg mass per structure floor. The error norm in each case is calculated as the ratio between the standard deviation of the displacement error and the simulated displacement. As shown in Figures 10- 12, the error norm decreases as the degrees-of-freedom increase, even

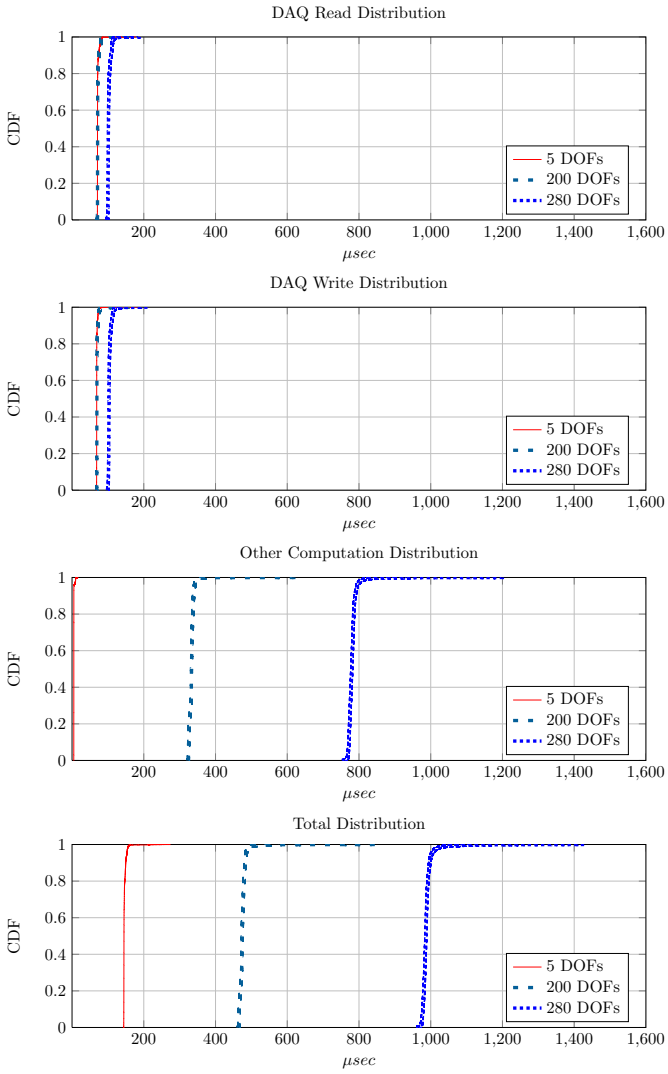



Figure 9: DAQ read/write and computation costs

though the 60 DOF case introduces more deadline misses.

This result is an interesting and important finding since it demonstrates that for some experiments there may be scenarios where more deadline misses are not always worse for the application. However, increasing numbers of deadline misses cause increasing desynchronization of cyber and physical components, which may result in incorrect command signals being issued to an actuator. The impact of these incorrect command signals is strongly linked to the characteristics of the specimen selected for testing and cannot fully be evaluated with a single example. It is clear that withing the same testing configuration, the overall test performance will deteriorate eventually as more and more deadline misses are introduced into the system. Thus, at least some bound on deadline misses is generally essential for this type of testing.

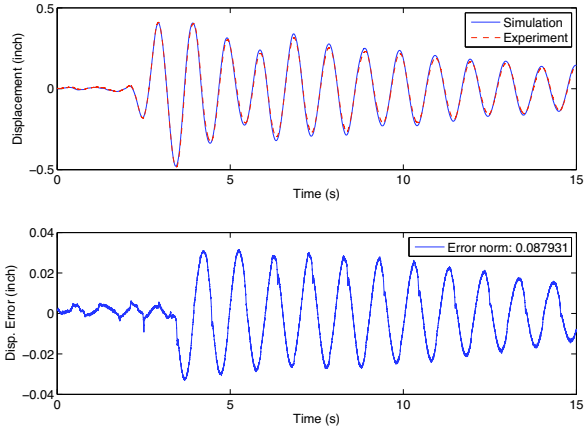



Figure 10: Single DOF comparison

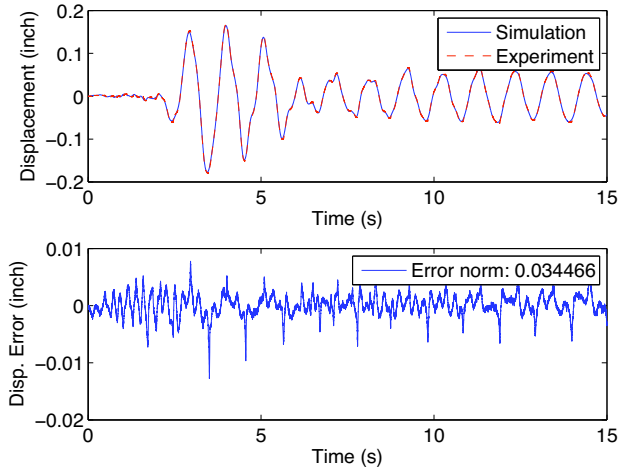



Figure 11: 5 DOF comparison

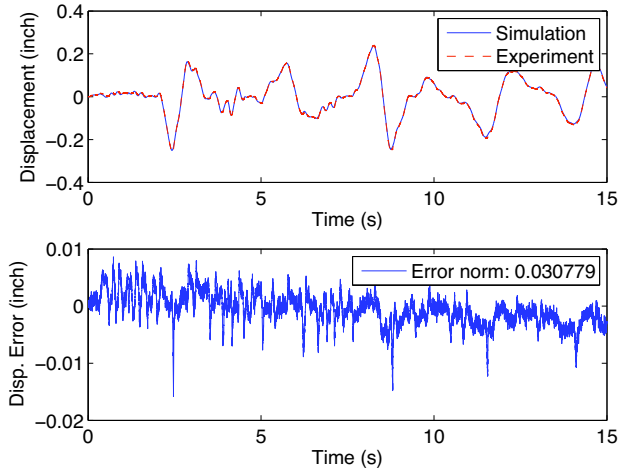



Figure 12: 60 DOF comparison

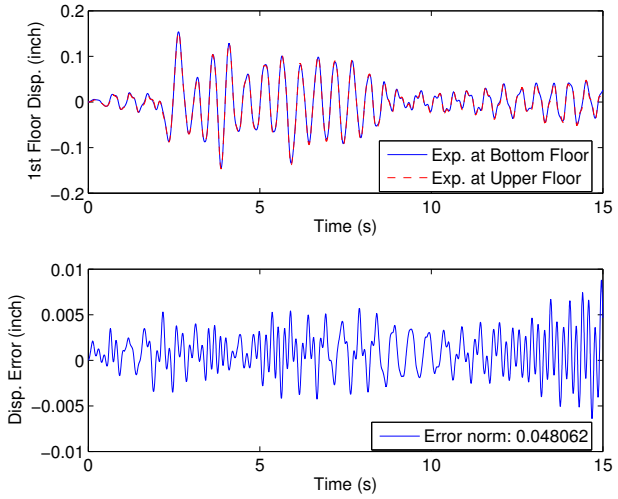



Figure 13: Top floor virtual, bottom floor physical

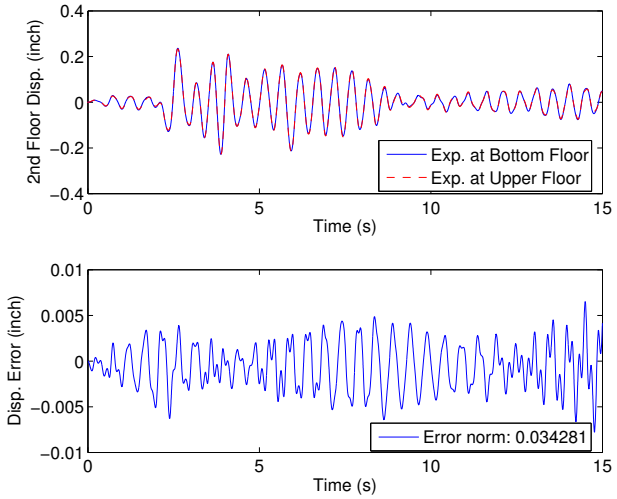



Figure 14: Top floor physical, bottom floor virtual

## 6.3 Interchanging Components

To evaluate further the effectiveness of our approach, we also tested a two-story shear building (mechanically simplified as a two degrees-of-freedom mass spring damper system) where half of the stiffness on each floor is assumed. A single physical actuator device is used, and the other device is virtualized (i.e., simulated along with the numerical substructure to form an internally closed-loop cyber-physical system). This virtualized component is assumed to be on top vs. bottom floor respectively (as illustrated in Figure 4) and the test results are compared in Figures 13 and 14. The good match between the curves seen in those figures indicates that the physical component is synchronized well with numerical component, the control method is effective, and that high-fidelity virtualization of either floor is feasible.

# 7. CONCLUSIONS AND FUTURE WORK

In this paper we have described how real-time hybrid testing motivates development of infrastructures that can address the kinds of requirements described in Section 2. In Section 4 we presented the design and implementation of novel middleware for a Cyber-physical Instrument for Real-time hybrid Structural Testing (CIRST) we are developing in this research. The case study in Section 5 and the experimental results in Section 6 demonstrate both the suitability of the CIRST middleware for real-time hybrid testing, and the need for further investigation into remaining sources of timing variability and other factors that may impact real-time hybrid test systems.

The data obtained from our real-time hybrid testing experiments matched reasonably well with the simulation results for all tests, which indicates that the infrequent deadline misses did not have a large impact on the fidelity of the hybrid testing experiments. The data also revealed new and relevant insights into real-time hybrid structural testing, such as the observations that (1) while the relative timing contribution of I/O between cyber and physical components decreased as the number of degrees of freedom increased in our experiments, the variation in I/O timing increased; and (2) a corresponding increase in deadline violations as the number of degrees of freedom increased did not impact fidelity of the system significantly. Both of these observations suggest that making similarly detailed measurements and comparisons is warranted for each particular testing configuration, since our results indicate the potential for trade-offs among cyber-physical properties of these systems.

While the results presented here demonstrate the feasibility of virtualizing different components, to make test specimens and test conditions more pluggable (and thus make testing even more flexible) further investigation is needed into how experiments can be *composed* using different modular structure, actuator, and compensation models. Further experimental case studies are needed as well, to evaluate the fidelity with which such composition can be achieved while maintaining real-time guarantees.

Since the variations between the hybrid tests and the simulations in the experiments presented in this paper could be attributed to a variety of possible sources including imperfect sensor measurement, modeling inaccuracy which does not consider non-linearity of the physical test specimen, or insufficiently detailed timing information about the physical specimen or the actuator, further study is also needed to characterize and quantify more fully the effects of deadline misses in the cyber portions of real-time hybrid testing systems.

We plan to expand the use of parallel execution of the numeric simulation algorithms, aided by data synchronization features of the CIRST middleware, to achieve and quantify further scalability of our approach through aggregation of more computational resources. We also plan to improve our hydraulic actuator models to capture the dynamics of physical components over a broader frequency range, along with associated non-linearity. In conjunction with further study of the effects of timing jitter, we plan to explore other dy-

namic compensation strategies in order to apply the desired signals to test specimens more accurately.

# 8. REFERENCES


[1] dSPACE systems. http://www.dspace.de/ww/en/inc/home.cfm.

[2] Ets. http://www.intervalzero.com/ets.htm.

[3] Metropolis: Design Environment for Heterogeneous Systems. http://embedded.eecs.berkeley.edu/metropolis/.

[4] The object management group. 2005. uml profile for modeling and analysis of real-time and embedded systems. omg request for proposals, real-time/05-02-06. http://www.omg.org/cgi-bin/doc?real-time/05-02-06.pdf.

[5] OpenSees, a software framework for developing applications to simulate the performance of structural and geotechnical systems subjected to earthquakes. http://opensees.berkeley.edu/.

[6] Ptolemy Project: Heterogeneous Modeling and Design. http://ptolemy.berkeley.edu/ptolemyII/.

[7] Suites of earthquake ground motions for analysis of steel moment frame structures: 10 pairs of horizontal ground motions for los angeles with a probability of exceedence of 10% in 50 years. http://nisee.berkeley.edu/data/strong_motion/sacsteel/.

[8] xPC target 4.1, perform real-time rapid prototyping and hardware-in-the-loop simulation using PC hardware. http://www.mathworks.com/products/xpctarget/.

[9] Working draft for the c++ language. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2008/n2800.pdf, 2009-3-23.

[10] M. Ahmadizadeh, G. Mosqueda, and A. Reinhorn. Compensation of actuator delay and dynamics for real-time hybrid structural simulation. *Earthquake Engineering and Structural Dynamics*, 37(11):21–42, 2008.

[11] J. Carrion and S. B.F. A model based delay compensation approach for real time hybrid testing. In $4^{th}$ *International Conference on Earthquake Engineering*, Taipei,Taiwan, 2006.

[12] J. Carrion and B. F. Spencer. Model-based strategies for real-time hybrid testing. Technical Report NSEL-006, University of Illinois at Urbana-Champaign, 2007.

[13] D. Group. nORB - Special Purpose Middleware for Networked Embedded Systems, 2005. deuce.doc.wustl.edu/nORB/.

[14] Institute for Software Integrated Systems. Component-Integrated ACE ORB. www.dre.vanderbilt.edu/CIAO/, Vanderbilt University.

[15] Institute for Software Integrated Systems. The ACE ORB. www.dre.vanderbilt.edu/TAO/, Vanderbilt University.

[16] S. Mahin, P. Shing, C. Thewalt, and R. Hanson. Pseudodynamic test method. Current status and future directions. *Journal of Structural Engineering*, 115(8):2113–2128, 1989.

[17] S. A. Mahin and P. B. Shing. Pseudodynamic method for seismic testing. *Journal of Structural Engineering*, 111(7):1482–1503, 1985.

[18] D. C. Schmidt. ACE: an Object-Oriented Framework for Developing Distributed Applications. In *Proceedings of the* $6^{th}$ *USENIX C++ Technical Conference*, Cambridge, Massachusetts, Apr. 1994. USENIX Association.

[19] P. B. Shing, Z. Wei, R. Y. Jung, and E. Stauffer. NEES fast hybrid test system at the University of Colorado. In $13^{th}$ *World Conference on Earthquake Engineering*, Vancouver, Canada, 2004.

[20] V. Subramonian, L.-J. Shen, C. Gill, and N. Wang. The design and performance of configurable component middleware for distributed real-time and embedded systems. In *RTSS '04: Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pages 252–261, Washington, DC, USA, 2004. IEEE Computer Society.

[21] V. Subramonian, G. Xing, C. Gill, C. Lu, and R. Cytron. Middleware specialization for memory-constrained networked embedded systems. In *Proceedings of 10th IEEE Real-time and Embedded Technology and Applications Symposium*, 2004.

[22] T. Tidwell, X. Gao, H.-M. Huang, C. Lu, S. Dyke, and C. Gill. Towards Configurable Real-Time Hybrid Structural Testing: A Cyber-Physical Systems Approach. In $12^{th}$ *International Symposium on Object-Oriented Real-time Distributed Computing*, Tokyo, Japan, Mar. 2009. IEEE.