CHAPTER 1

Simulation and Validation of Structural Models

Christoph Hoffmann, Ahmed Sameh, and Ananth Grama
Department of Computer Science,
Purdue University.
{cmh, sameh, ayg}@cs.purdue.edu

There has been significant recent work on development of simulation methodology and validation techniques for understanding the behavior of large structures – buildings, bridges, dams, etc. These efforts are aimed at designing structures that are robust to natural excitations (earthquakes, floods, high winds) as well as human factors (explosions, fire). The eventual goal of these efforts is to understand why structures fail and how to mitigate these failures. In this chapter, we describe our recent efforts aimed at developing complex structural models, using them to understand structural failure (the 9/11 collapse of the Twin-Towers), and experimental efforts aimed at calibrating and validating computational models. In particular, we demonstrate that using high-performance computing platforms, powerful simulation engines, high-resolution geometric models, and accurate materials models, we can gain considerable understanding of structural response to excitations. We also describe the complexity of experiment design and data acquisition from experiments.

1.1 Simulation of Large-Scale Structures

The objective of computational simulation of structures is to develop models capable of describing structural response to a high degree of accuracy. These models are used to predict failure modes, as well as to guide techniques for model reduction. Reduced-order models (models with fewer degrees of freedom) can be used for real-time control of structures. In this chapter, we describe our work on model construction, simulation, and validation. We also present some insights on the interplay between the numerical and geometric aspects of the problem.

The initial model for the simulation has two parts: a description of the geometry of the elements and a description of the physical properties. The former is translated into node coordinates and connectivity information to specify the elements (beam, shell, and volume elements), the latter reduces to a system of equations from which a numerical problem is constructed. Since high-fidelity models of structures have large numbers of elements, it is best to use electronic models where available, and to discretize them using an automatic mesh generator, instead of building finite-element models from scratch. Many choices for mesh generation exist, with varying degrees of suitability for classes of objects. The annual International Meshing Roundtable [9] is an excellent venue for familiarizing ones self with the latest tools and techniques in meshing. We describe each of these phases of modeling using a case study of the 9/11 crashes.

1.1.1 Modeling the North Tower of the World Trade Center and the Airframe

In the case of the World Trade Center simulation, we were able to use a partially meshed model of the exterior of the North Tower (WTC-1) which was then completed by adding the interior structure, consisting of the buildings core, the spandrel beams supporting the floor and the elements for the concrete flooring as well. Staircases were not modeled in detail. This task of completing the interior of WTC-1 used public domain blueprints. The model had 640,000 nodes, 530,000 beam elements and 360,000 shell elements.

Aside from the model size, meshing the building structure of modern high-rises is not difficult. However, much of the detailed structure of the joints gets abstracted. By this, we mean that the geometric structure is reduced to a shared node among several beam elements, say. The corresponding material response is then captured by the governing equations and material properties. Also, the beam shapes are often abstracted as well, represented simply by line elements with an intrinsic orientation and a specification of the type of beam. This geometric information is lost in translation and has to be reconstructed for visualizing the simulation results. Figure 1.1 illustrates this issue. Fortunately, obtaining reliable information about the WTC-1 building was not difficult, and the validity of the model is established through experiments, described later in this chapter.

Modeling the airplane posed challenges, however. Here, it turned out to be difficult to obtain reliable information about the Boeing 767 airframe structure. Overall dimen-

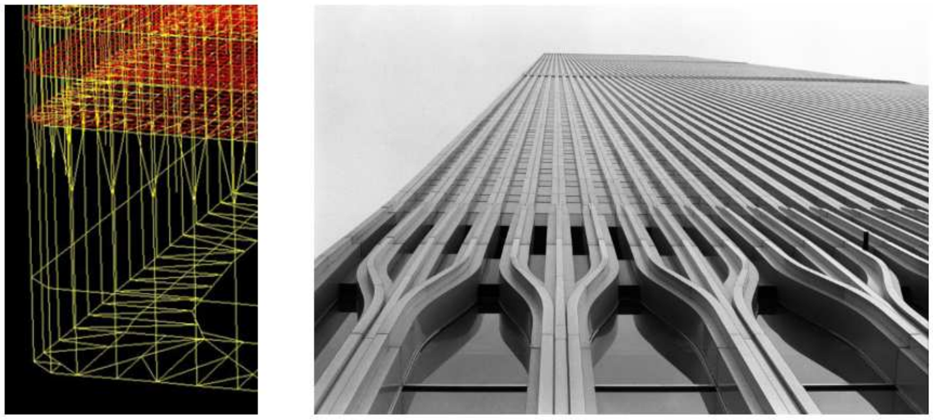

Figure 1.1 Structure of ground floor facade for the FEA and corresponding geometry. The FEA abstraction cannot properly represent the complexity of the shapes, especially at their meeting points.

sions are readily available, but except for an accurate reconstruction of the planes exterior, little information is available about the interior structure, the shape and dimensions of the beams, ribs, and stringers. Understandably, Boeing considers a detailed model of its planes proprietary, so we needed to find a different way to create a reasonable model [2] and test its accuracy as best as possible. We undertook this task beginning with a graphical model of the plane's exterior.

A graphical model is wholly unsuitable for finite element analysis (FEA). Its surfaces are often composed of very long and slender triangles; a minimum angle of less than 10 degrees is not uncommon. Moreover, relying on the graphics hardware to cull invisible regions of the surface, parts such as the skin of a wing will extend part-way into the interior of the fuselage without any explicit representation of the intersection curve. Thus, the first step is to rectify the airplane's skin and to suitably mesh it, a task that requires substantial manual labor even using the meshing capabilities of LS-Dyna [4]. We then added to the fuselage ribs and stringers, the keel beam, the wing anchor beams, floor supports and the floor itself. We also added spars and ribs to the wings, aileron and rudder. Aircraft engines and landing gear contain substantial parts. The landing gear typically is a titanium forging, and the engine shaft is another substantial part. Both structures were modeled and meshed appropriately. Sections of the resulting mesh are shown in Figure 1.2.

1.1.2 Model Calibration

Since there is little hard data on the structural components of the aircraft, we consulted experts in aerospace engineering and used the Riera approach to calibrate the aircraft model [14, 15]. The Riera approach is an upshot of experiments at Sandia National Labs measuring reaction forces and damage from aircraft impact. The impact forces, measured from

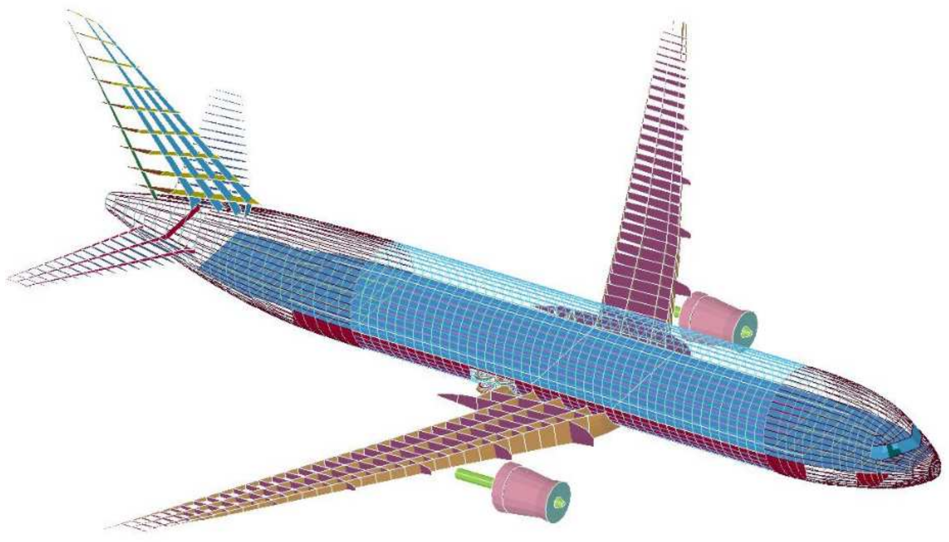

Figure 1.2 Meshed structural elements of the aircraft. The main landing gear is partly visible behind the wing.

the time of impact to its completion, relate intuitively to the ability of the impacting structures to absorb energy by deformation. Accordingly, the forces are highest when the most massive and rigid structural elements hit. Figure 1.3 shows both a moment in our impact simulation and the Riera curve used to calibrate the aircraft model.

The fuel of the impacting aircraft carries a large fraction of the kinetic energy. We modeled the fuel using smoothed particle hydrodynamics (SPH), which benefited from a separate calibration step. The SPH calibration was done by our collaborators in Civil Engineering modeling a beverage can impacting a rigid metal plate. Led by Santiago Pujol, the simulation results were compared with experimental measurements. Both are shown in Figure 1.4.

1.1.3 Simulation and Results

After constructing those models and calibrating them in the indicated manner, we then simulated the impact of the loaded airframe into the top floors of the WTC-1 building. Initial conditions were chosen to approximate the known attitude and position of the North Tower impact.

The difficulty obtaining accurate dimensions and properties of the air frame make quantitative conclusions from the simulations difficult. In fact, there are widely diverging estimates of the core damage of the North tower [3, 8, 13, 16]. Nevertheless, it is possible

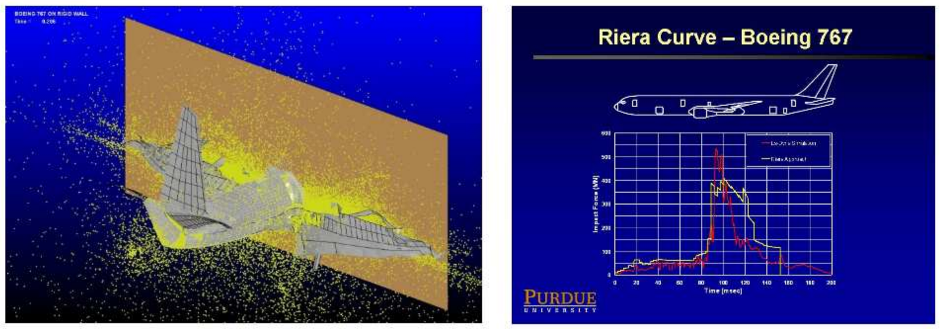

Figure 1.3 Impact of our Boeing 767 model into a rigid wall, yielding the yellow curve on the right. Predicted curve based on the literature in red. Fuel in the tanks is modeled using smoothed particle hydrodynamics (SPH).

to conclude with considerable confidence that the collapse mechanism of the North Tower was initiated by the ensuing fires [7]. Those fires easily elevated the temperature of the core structure sufficiently so that it could not carry the building load. That load, about 60undamaged building, was increased by the extra load that rested on the core due to the perimeter structure being cut in a wide swath. Further corroboration that our simulation captured the essential elements of the event, despite the uncertainty surrounding the exact properties of the air frame, can be seen in Figure 1.5, which compares the simulated damage to the facade, at the entry of the colliding aircraft, with a contemporary image reproduced from [12].

1.1.4 Visualization of Simulation Results

The final part of the 9/11 simulation effort, led by our colleague Voicu Popescu, was to render and animate the results of the simulation using state-of-the-art techniques from computer graphics. Downloaded to-date more than 5.5M times, the simulation has generated tremendous debate and continues to be used in documentaries in the US and abroad. Here, the key step was to export the simulation data, as computed by the FEA system, and import it into 3dsMax, an animation system that incorporates leading-edge graphics algorithms. In this part of the effort, a major difficulty is the aforementioned abstraction of geometric shape, particularly the shape of beams and how different structural elements are joined. Some still images from the visualization are shown in Figure 1.6.

Model Reduction. Reducing a model such as the WTC-1 building and the impacting aircraft can be done by simplifying the geometry and also by simplifying the physical model. Geometry simplification is quite common in finite element analysis: symmetry is exploited where possible; in the case of WTC-1, details of the lower floors of the building are eliminated; detail shapes such as beam interconnections and cross sections are not

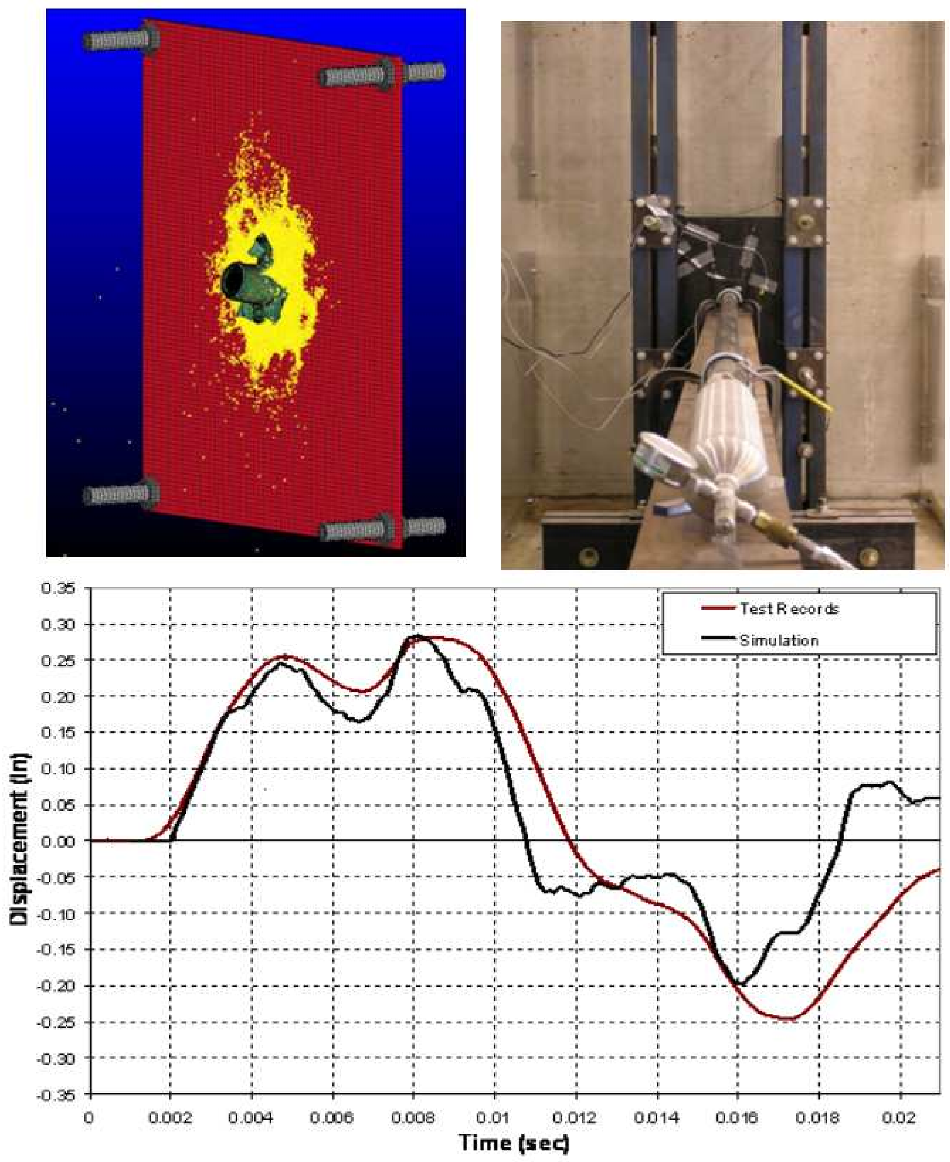

Figure 1.4 Simulated beverage can impact (top left) and experimental setup (top right). Resulting time-displacement curve (bottom). Measured values in red, simulation prediction in black. Work by Santiago Pujol, Civil Engineering, Purdue University.

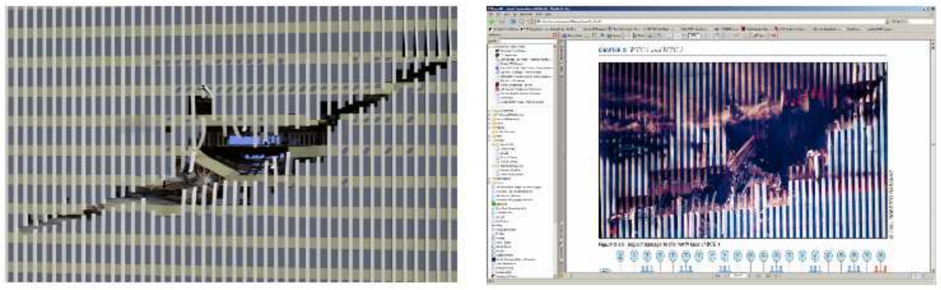

Figure 1.5 Facade damage determined by the simulation (left) compared with a contemporary photograph [12] (right).

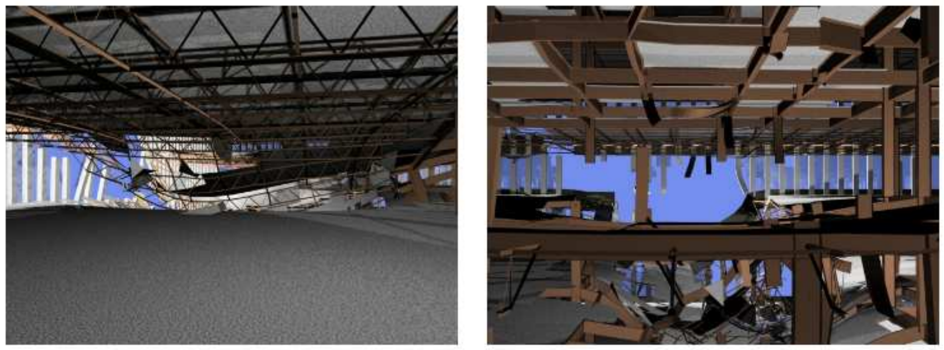

Figure 1.6 Damage to Floors 95 and 96 (left); reverse angle shot from floor 96 (right). Note the detailed geometry of the spandrel beams and core beams represented by line elements in the simulation.

represented geometrically. Simplifying the physics based on the numerical characteristics of the problem is novel and promising. It is a topic of intense research efforts, and deserves a comprehensive treatise by itself. Since this kind of simplification seeks to reduce the degrees of freedom of the problem, it also implies a large-scale geometry simplification. As we have seen, such simplifications of the geometry lose important detail that will be missed when creating subsequently sophisticated visualizations. In the larger scientific work-flow, therefore, it is helpful to have a detailed geometric model from which to derive, as suggested by the numerical properties, a low-order shape approximation that can be lifted back to the original, detailed geometry model. Both the reduced-order physics and the corresponding shape model, therefore, become information structures derived from a master model, a concept familiar to the community from discrete manufacturing.

1.2 Calibration and Validation of Structural Modeling

Two important aspects of computational simulations are, calibration of underlying models (material properties, joint stiffness, etc.), and validation of models using measured excitations and structural response. We refer to some of these above – namely, the Riera curve approach and the SPH calibration through fluid-structure impaction. Experimental validation requires instrumentation of structures, real-time techniques for data acquisition, and data analysis. Each of these represent significant computational challenges associated with implementation of the complex distributed software that senses, communicates, and stores data. This distributed program has exacting requirements on performance (real-time), constraints on resources (communication rates, buffers), and guarantees on correctness. In heterogeneous fault-prone environments such as structures subject to external stresses, implementing such distributed programs is a major undertaking. In the rest of this chapter, we describe efforts aimed at implementing real-time sensing and control.

As applications of sensor networks mature, there is increasing realization of the complexity associated with programming large numbers of heterogeneous devices operating in highly dynamic environments. Much of this complexity stems from the need to account for network state, failures, time and resource constraints, heterogeneity and scalability issues, and data-driven in-network processing. Even tasks that are conceptually straightforward from the point of view of aggregate system specification, such as data acquisition, processing, and aggregation require significant programming effort.

Traditional approaches to programming sensor networks rely on "network-enabled" applications for individual nodes that code distributed behavior through explicit messaging. Application development in this framework often requires implementation of system-level primitives, or reliance on inflexible or inefficient underlying platforms. Consequently, application development is time- and effort-intensive, error-prone, and has limitations with respect to scalability, heterogeneity, and performance. Indeed, software development cycles for complex applications at current state-of-the-art do not keep pace with updates in sensing hardware. In contrast, a macroprogram directly specifies the behavior of a distributed ensemble. Through suitable abstractions, it eases programmer burden in dealing with resource constraints, performance optimization, and scalability and adaptability w.r.t. to network and load dynamics. Macroprograms can often be statically verified to enhance robustness and to enforce time and resource constraints.

The presence of a small number of low-cost and low-power, yet relatively powerful devices (such as X-Scale based Intel Stargate devices in a network of Mica motes) can significantly enhance sensor network performance [17]. Existing development platforms target development for specific device capabilities – for example, mote-scale devices (e.g., [6, 5]), or networks of higher performance machines in a sensor network (e.g., [10]). This lack of vertical integration across heterogeneous devices results in increased development complexity and programmer effort. Consequently, there is a need for a unified macroprogramming model that realizes benefits from heterogeneous environments. The dynamic nature of such networks necessitates a runtime environment that can execute on resource-constrained motes, such as Mica2, while potentially scaling to resource rich nodes.

The goal of supporting easy-to-use high-level abstractions that hide system-level details is often at odds with requirements of low overhead and flexibility. This tradeoff is the primary determinant of the design choices associated with a realizable macroprogramming architecture. The underlying architecture must provide a low overhead execution environment for fundamental macroprogram primitives, while naturally supporting rich extensions based on the system's operational requirements or application domain. Among other design objectives for a macroprogramming architecture, the reuse of software components across applications is an important engineering parameter, specially as sensor systems become more commonplace.

With these motivating objectives, we have developed COSMOS, an architecture for macroprogramming heterogeneous sensor networks. COSMOS is comprised of a lean operating system, mOS, and an associated programming language, mPL. COSMOS supports complex applications, built upon reusable components. In the COSMOS framework, aggregate system behavior is specified in terms of the underlying distributed data processing. This specification consists of functional components (FCs), which provide computing primitives, and an interaction assignment (IA), which specifies distributed dataflow through FCs. An FC specifies a typed declaration in mPL and the corresponding function, specified in a suitably constrained subset of C. Node capability constraints associated with each FC allow the programmer to effectively and safely utilize heterogeneous devices in the network.

mPL supports the expression of IAs as a fundamental primitive with support for static program verification. Synergistically, mOS provides a low-footprint runtime environment providing dynamic instantiation of FCs and realization of IAs through these instances, along with other requisite OS subsystems. Building on this simple notion of expressing behavior through composition of FCs, the COSMOS architecture supports the ability to easily append rich high-level macroprogramming abstractions in mPL, without modification to the OS.

1.2.1 COSMOS Runtime Environment

The fundamental design objective for our runtime system is to minimize its processing and memory footprint, and to allow performance scaling on resource rich nodes. A macroprogram specifies connections between FCs and device port instances using abstract asynchronous data channels. Abstract data channels are realized as data queues, whose nodes encapsulate data. Each device port of an FC instance has an individual queue associated with each of its outputs. During application initialization, these queues are attached to the inputs of an instance of the next component, as specified by the IA. The runtime system also handles network data flow transparently. If the communicating instances are not on the same node, output queues connect to the network service FC of the source nodes. An output queue from the network service of the destination node to the succeeding component completes the virtual data channel. The asynchronous semantics ideally suit the network

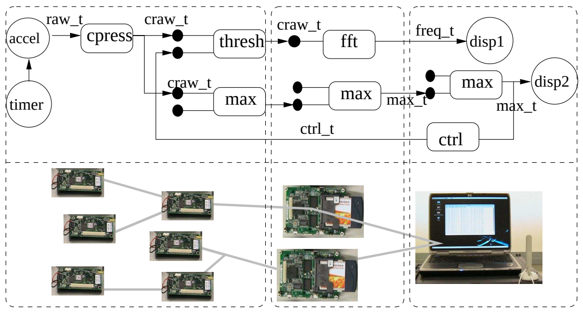

Figure 1.7 A macroprogram and its instantiation. This program displays maximum acceleration values for a structure, evaluates frequency response of the structure, and displays the resulting spectrogram if the acceleration is above a specified threshold. A controller (ctrl) feeds the threshold value back, based on aggregate data from the network. This conserves system resources until an interesting event triggers the need for a high fidelity view of the data.

data channel and impose few restrictions on the semantics of the underlying network communication, allowing the use of simple low overhead protocols.

1.2.2 mOS Operating System Architecture

The *mOS* operating system provides a low overhead implementation of the runtime system for COSMOS based macroprogramming. Architecturally, mOS consists of a core kernel, which is logically divided into a platform independent core and hardware specific drivers and routines. The key subsystems of the platform independent core include the scheduler, timer, dynamic memory manager, dynamic component loading and dataflow setup manager, dataflow API, device abstraction layer, and a system information and control API. We have implemented the mOS operating system on Mica2 and POSIX platforms. On motes, mOS is a full-fledged operating system directly running atop the underlying hardware. On resource rich nodes (e.g., in our case, Stargate SBC devices and PCs running Linux), mOS sits atop the POSIX layer, and provides a transparent environment for executing macroprogram applications, which are, by design, platform independent. To execute a macroprogram on POSIX devices, the mOS management thread loads FCs (which may be a part of the binary of mOS or compiled as individual loadable libraries), and executes the FCs (waiting for inputs) each as an individual thread. Both on POSIX, and on the motes, mOS is accessible to the dynamically loaded components through a system call pointer table.

COSMOS is currently in use on a real-world three-story building test-bed to study structural response to ground motion, at the Bowen Labs [1] at Purdue University (Figure 1.8).

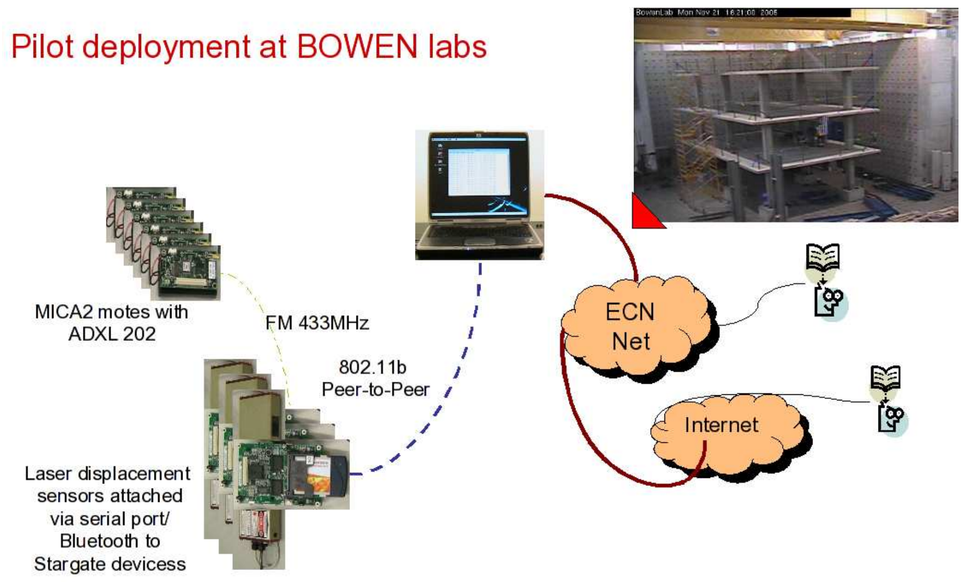
Pilot deployment at BOWEN labs
MICA2 motes with ADXL 202
FM 433MHz
802.11b Peer-to-Peer
Laser displacement sensors attached via serial port/ Bluetooth to Stargate devicess
ECN Net
Internet

Figure 1.8 Deployment and testing of structural response using the COSMOS macroprogramming infrastructure.

1.3 Concluding Remarks

Simulation and validation of structural models are critical components of understanding and designing robust structures. Simulations require complex geometric models, material property databases, high-performance simulation engines, and massively parallel computers. Tests on real structures serve two important purposes – they provide material properties and model calibration for simulations and validate simulation results through experiments. Comprehensive validation requires complex instrumentation, multimodal sensing, and real-time data acquisition, typically through wireless sensor networks. Significant

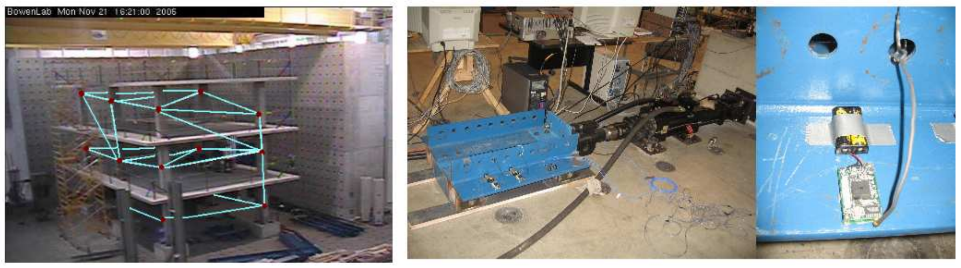

Figure 1.9 Operational tests – electrohydraulic rams excite the structure and response is measured through a variety of self-organizing sensors.

progress has been made in all of these computational aspects – leading to real-life deployment of many of these technologies in design and operation.

Acknowledgments

The collaboration with Profs. Mete Sozen, Santiago Pujol, Ayhan Irfanoglu, and Voicu Popescu are gratefully acknowledged. The work has been partially supported by NSF Grant DSC-0325227 and by the Northwest Indiana Computational Grid.

Bibliography

[1] Bowen labs. `https://engineering.purdue.edu/CE/BOWEN/Facilities`.

[2] M. Badrocke and B. Gunston. *Boeing Aircraft Cutaways*. Osprey Publishing Ltd., Oxford, 1998.

[3] Z. P. Bazant and Y. Zhou. Why did the world trade center collapse? *Journal of Engineering Mechanics*, 128(1) 2-6, 2002.

[4] Livermore Software Corporation. Ls-dyna, 2009.

[5] Chih-Chieh Han, Ram Kumar Rengaswamy, Roy Shea, Eddie Kohler, and Mani Srivastava. SOS: a dynamic operating system for sensor networks. In *Proc. of MobiSys '05*, June 2005.

[6] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *Proc. of ASPLOS-IX*, November 2000.

[7] A. Irfanoglu and C. Hoffmann. An engineering perspective of the collapse of wtc-1. *ASCE Journal on Performance of Constructed Facilities*, 22,62, 2008.

[8] M. Karim and M. Fatt. Impact of the boeing 767 aircraft into the world trade center. *Journal of Engineering Mechanics*, 131(10) 1066-1072, 2005.

[9] Sandia Laboratories. International meshing roundtable, 2009.

[10] Ting Liu and Margaret Martonosi. Impala: a middleware system for managing autonomic, parallel sensor systems. In *Proc. of PPoPP '03*, June 2003.

[11] Bertrand Meyer. Applying design by contract. *IEEE Computer*, 25(10):40–51, October 1992.

[12] National Institute of Standards and Technology (NIST). Final report of the national construction safety team on the collapses of the world trade center towers, nist ncstar 1, 2005.

[13] Y. Omika, E. Fukuzawa, N. Koshika, H. Morikawa, and R. Fukuda. Structural responses of world trade center under aircraft attacks. *Journal of Structural Engineering*, 131(1), 6-15, 2005.

[14] J. D. Riera. On the stress analysis of structures subjected to aircraft impact forces. *Nuclear Engineering and Design*, 8:415-426, 1968.

[15] T. Sugano, H. Tsubota, Y. Kasai, N. Koshika, H. Ohnuma, W. von Riesemann, D. Bickel, and M. Parks. Full-scale aircraft impact test for evaluation of impact force. *Nuclear Engineering and Design*, 140:373-385, 1993.

[16] T. Wierzbicki and X. Teng. How the airplane wing cut through the exterior columns of the world trade center. *International Journal of Impact Engineering*, 2003.

[17] Mark Yarvis, Nandakishore Kushalnagar, Harkirat Singh, Anand Rangarajan, York Liu, and Suresh Singh. Exploiting heterogeneity in sensor networks. In *Proc. of INFOCOM '05*, March 2005.