# Academic Challenges in Large-Scale Multiphysics Simulations*

Michael T. Heath and Xiangmin Jiao

Computational Science and Engineering,
University of Illinois, Urbana, IL 61801, USA
{heath, jiao}@cse.uiuc.edu

**Abstract.** Multiphysics simulations are increasingly playing a critical role in scientific and engineering applications. The complex and cross-disciplinary nature of such applications poses many challenges and opportunities in both research and education. In this paper we overview some of these research challenges, as well as an academic program designed to prepare students to meet them.

## 1 Introduction

Many physical and biological systems of interest today involve multiple interacting components with diverse space and time scales, diverse material properties, and many other sources of heterogeneity. Modeling and simulation of such systems is particularly challenging because of the diversity of knowledge and techniques required, which in turn poses a severe challenge to conventional educational programs whose compartmentalization often discourages the necessary breadth. In this paper we overview some of the research challenges arising in a fairly typical research project involving large-scale multiphysics simulations, and we also discuss an educational program designed to provide students with the cross-disciplinary expertise necessary to address such challenges successfully.

Not coincidentally, both the research project and the educational program we will describe are part of an integrated organizational framework, called Computational Science and Engineering, that has evolved at the University of Illinois as both an educational program and a research program. The goal of the educational program is to produce computationally literate scientists and engineers on the one hand, and applications-aware computer scientists on the other. Students in this educational program become "bilingual," learning the language of computing as well as the language of one or more application disciplines, such as physics, chemistry, biology, materials science, or engineering. A major goal of the research program is to enable students to experience while still on campus the kind of cross-disciplinary, team-oriented research that they are being prepared to engage in after graduation.

* Research supported by the U.S. Department of Energy through the University of California under subcontract B523819.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 52–59, 2005.

## 2 Computational Science and Engineering

At Illinois, Computational Science and Engineering (CSE) is an interdepartmental program encompassing fourteen participating departments. Students in the program receive a conventional degree (M.S. or Ph.D.) from one of these departments and also a certificate of completion of the CSE option, which is in effect analogous to a graduate minor. Core courses for the CSE option include data structures and software principles, numerical analysis, parallel programming, and scientific visualization. More advanced courses include parallel numerical algorithms, parallel computer architecture, computational mechanics, computational physics and materials science, and advanced finite element methods. Many of these courses are specifically designed to be accessible to students from multiple departments. The specific courses chosen, and the number of courses required, depend on the home department of the student, but the requirements are reasonably uniform across departments.

The courses provided by CSE play an important role in developing the breadth of expertise necessary to do true interdisciplinary research, but courses alone do not suffice. To enable students to gain first-hand experience with large-scale computation, CSE also provides computational facilities for both research and class projects. These facilities include a workstation laboratory, symmetric multiprocessor servers for shared-memory programming, and, most importantly, a very large cluster for distributed-memory programming. The current cluster owned and operated by CSE features 640 Apple Xserves—each with two G5 processors, for a total of 1280 processors—interconnected by a high-bandwidth, low-latency Myrinet network. A cluster of this size enables our students, faculty, and staff to solve very large problems and perform simulations at a scale comparable to those at the largest laboratories and supercomputer centers.

Courses can provide the relevant knowledge, and computing facilities can provide the requisite computational power, but there is still no replacement for the experience of actually *applying* them to concrete problems in science and engineering. To provide such experience, CSE also hosts multiple interdisciplinary research projects. One of these, the NSF-funded Center for Process Simulation and Design, focuses on modeling and simulation of industrial processes, such as casting and extrusion, with complex and evolving geometries. The largest project hosted by CSE is the Center for Simulation of Advanced Rockets (CSAR), funded by DOE's Advanced Simulation and Computing program, whose major goals are to advance the state of the art in computational simulation of complex systems and to train a new generation of computational scientists and engineers, which conveniently coincide with the goals of CSE. Next we will briefly describe CSAR's research program.

The principal objective of CSAR is to develop an integrated software system, called *Rocstar*, for detailed whole-system simulation of solid rocket motors [2], such as the Space Shuttle Reusable Solid Rocket Motor (RSRM) illustrated in Fig. 1. An overview of the components of the current generation of *Rocstar* is shown in Fig. 2. This system involves three broad physics disciplines—fluid dynamics, solid mechanics, and combustion—that interact with each other at the

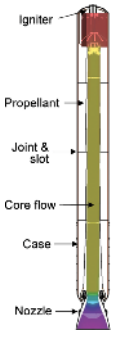


**Fig. 1.** Schematic of RSRM

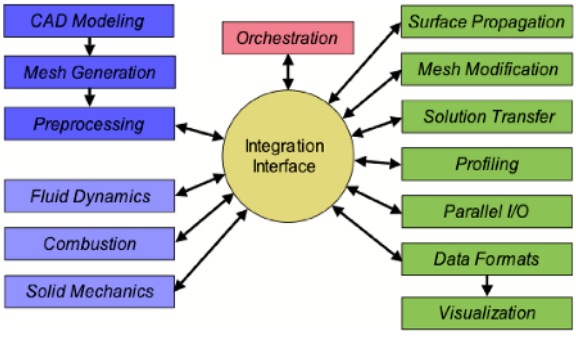


**Fig. 2.** Overview of *Rocstar* software components

primary system level. In addition, there are subsystem level interactions, such as particles and turbulence within fluids. The coupling issues associated with these complex interactions are representative of a large class of multiphysics simulations.

Effective coupling of multiphysics simulations poses challenges and opportunities in many areas. We will focus on two broad areas—system integration and computational mathematics—that depend crucially on two core areas of the CSE educational program, respectively software engineering and numerical analysis.

## 3 System Integration

Because of their complex and cross-disciplinary nature, multiphysics systems are intrinsically demanding, requiring diverse backgrounds within the research team. They are particularly challenging when the individual physics components are not merely implementations of established technologies, but are at the frontier of their respective research agendas. Moreover, component coupling schemes are challenging research subjects in their own right.

To accommodate rapidly changing and evolving systems, a software framework must allow the individual components to be developed as independently as possible, and integrate them subsequently with few or no changes. It must provide maximum flexibility for physics codes and be adapted to fit the diverse needs of the components. These requirements are at odds with many traditional software architectures and frameworks, which typically assume that the framework is fully in control, and are designed for extension instead of integration.

### 3.1 Component Architecture

To facilitate the diverse needs of different components, we have developed an unconventional, asymmetric architecture in which software components are grouped into the following categories:

- *Physics modules* solve physical problems in their respective geometric domains. They are similar to stand-alone applications and typically written in Fortran 90 using array based data structures encapsulated in derived types.
- *Service modules* provide specific service utilities, such as I/O, communication, and data transfer. They are typically developed by computer scientists but driven by the needs of applications, and are usually written in C++.
- *Integration interface* provides data management and function invocation mechanisms for inter-module interactions.
- *Control* (*orchestration*) *modules* specify overall coupling schemes. They contain high-level domain-specific constructs built on top of service modules, provide callback routines for physics modules to obtain boundary conditions, and mediate the initialization, execution, finalization, and I/O of physics and service modules through the integration interface.

In *Rocstar*, the above categories correspond to the components at the lower-left, right, center, and top of Fig. 2, respectively. In addition, our system uses some *off-line tools*, such as those at the upper-left corner of Fig. 2, which provide specific pre- or post-processing utilities for physics modules.

### 3.2 Data Management

To facilitate interactions between modules, we have developed an object-oriented, data-centric integration framework called *Roccom*. Its design is based on *persistent objects*. An object is said to be *persistent* if it lasts beyond a major coupled simulation step. In a typical physics module, especially in the high-performance regime, data objects are allocated during an initialization stage, reused for multiple iterations of calculations, and deallocated during a finalization stage. Therefore, most objects are naturally persistent in multiphysics simulations.

Based on the assumption of persistence, *Roccom* defines a registration mechanism for data objects and organizes data into distributed objects called *windows*. A window encapsulates a number of *data attributes*, such as mesh (coordinates and connectivities), and some associated field variables. A window can be partitioned into multiple *panes* to exploit parallelism or distinguish different material or boundary-condition types. In a parallel setting, a pane belongs to a single process, while a process may own any number of panes. A module constructs windows at runtime by creating attributes and registering their addresses. Different modules can communicate with each other only through windows, as illustrated in Figure 3.

The window-and-pane data abstraction of *Roccom* drastically simplifies inter-module interaction: data objects of physics modules are registered and organized into windows, so that their implementation details are hidden from the framework and need not to be altered extensively to fit the framework. Service utilities can now also be developed independently, by interacting only with window objects. Window objects are self descriptive, and in turn the interface functions can be simplified substantially, frequently reducing the number of functions or the number of arguments per function by an order of magnitude. The window

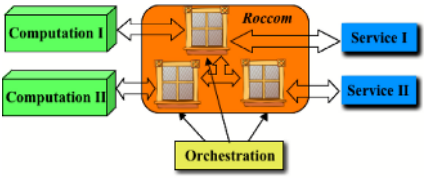


**Fig. 3.** Schematic of windows and panes

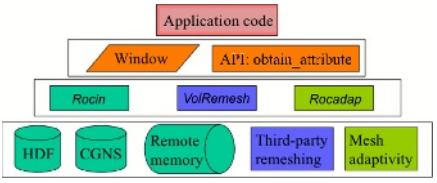


**Fig. 4.** Abstraction of data input

abstraction can be used for all data exchanges of a module, whether the other side is a service utility, files of various formats, or remote machines. For example, as illustrated in Fig. 4, file I/O services map *Roccom* windows with scientific file formats, and application modules obtain data from an input window through a generic function interface.

*Roccom* also introduces a novel concept of *partial inheritance* of windows to construct a sub-window by *using* or *cloning* a subset of the mesh or attributes of another window. In addition, the registered attributes in *Roccom* can be referenced as an aggregate, such as using "mesh" to refer to a collection of nodal coordinates and element connectivities. These advanced features allow performing complex tasks, such as reading or writing data for a whole window, with only one or two function calls. For more information on the novel features of *Roccom*, see [3].

## 4 Computational Mathematics

In *Rocstar*, a physical domain is decomposed into a volume mesh, which can be either block-structured or unstructured, and the numerical discretization is based on either a finite element or finite volume method. The interface between fluid and solid moves due to both chemical burning and mechanical deformation. In such a context, we must address numerous mathematical issues, three of which we discuss here.

### 4.1 Meshing-Related Issues

In *Rocstar*, each physics module operates on some type of mesh. A critical issue in integrated rocket simulations is the degradation of mesh quality due to the changing geometry resulting from consumption of propellant by burning, which causes the solid region to shrink and the fluid region to expand, and compresses or inflates their respective meshes. This degradation can lead to excessively small time steps when an element becomes poorly shaped, or even outright failure when an element becomes inverted. To address this issue, we take a three-tiered approach, in increasing order of aggressiveness: mesh smoothing, mesh repair, and global remeshing.

Mesh smoothing copes with gradual changes in the mesh. We provide a combination of in-house tools and integration of external packages. Our in-house effort

focuses on feature-aware surface mesh smoothing, and provides novel parallel algorithms for mixed meshes with both triangles and quadrilaterals. To smooth volume meshes, we adapted the serial MESQUITE package [1] from Sandia National Laboratories, parallelizing it by leveraging our across-pane communication abstractions.

If the mesh deforms more substantially, then mesh smoothing becomes inadequate and more aggressive mesh repair or even global remeshing may be required, although the latter is too expensive to perform very frequently. For these more drastic measures, we currently focus on tetrahedral meshes, using third-party tools off-line, including Yams and TetMesh from Simulog and MeshSim from Simmetrix. We have work in progress to integrate MeshSim into our framework for on-line use. Remeshing requires that data be mapped from the old mesh onto the new mesh, for which we have developed parallel algorithms to transfer both node- and cell-centered data accurately.

### 4.2 Data Transfer

In multiphysics simulations, the computational domains for each physical component are frequently meshed independently, which in turn requires geometric algorithms to correlate the surface meshes at the common interface between each pair of interacting domains to exchange boundary conditions. These surface meshes in general differ both geometrically and combinatorially, and are also partitioned differently for parallel computation. To correlate such interface meshes, we have developed novel algorithms to constructs a *common refinement* of two triangular or quadrilateral meshes modeling the same surface, that is, we derive a finer mesh whose polygons subdivide the polygons of the input surface meshes [5]. To resolve geometric mismatch, the algorithm defines a *conforming homeomorphism* and utilizes locality and duality to achieve optimal linear time complexity. Due to the nonlinear nature of the problem, our algorithm uses floating-point arithmetic, but nevertheless achieves provable robustness by identifying a set of consistency rules and an intersection principle to resolve any inconsistencies due to numerical errors.

After constructing the common refinement, data must be transferred between the nonmatching meshes in a numerically accurate and physically conservative manner. Traditional methods, including pointwise interpolation and some weighted residual methods, can achieve either accuracy or conservation, but none could achieve both simultaneously. Leveraging the common refinement, we developed more advanced formulations and optimal discretizations that minimize errors in a certain norm while achieving strict conservation, yielding significant advantages over traditional methods, especially for repeated transfers in multiphysics simulations [4].

### 4.3 Moving Interfaces

In *Rocstar*, the interface must be tracked as it regresses due to burning. In recent years, Eulerian methods, especially level set methods, have made significant advancements and become the dominant methods for moving interfaces [7, 8]. In

our context Lagrangian representation of the interface is crucial to describe the boundary of volume meshes of physical regions, but there was no known stable numerical methods for Lagrangian surface propagation.

To meet this challenge, we have developed a novel class of methods, called *face-offsetting methods*, based on a new *entropy-satisfying Lagrangian* (*ESL*) *formulation* [6]. Our face-offsetting methods exploit some fundamental ideas used by level set methods, together with well-established numerical techniques to provide accurate and stable entropy-satisfying solutions, without requiring Eulerian volume meshes. A fundamental difference between face-offsetting and traditional Lagrangian methods is that our methods solve the Lagrangian formulation face by face, and then reconstruct vertices by constrained minimization and curvature-aware averaging, instead of directly moving vertices along some approximate normal directions. Fig. 5 shows a sample result of the initial burn of a star grain section of a rocket motor, which exhibits rapid expansion at slots and contraction at fins. Our algorithm includes an integrated node redistribution scheme that is sufficient to control mesh quality for moderately moving interfaces without perturbing the geometry. Currently, we are coupling it with more sophisticated geometric and topological algorithms for mesh adaptivity and topological control, to provide a more complete solution for a broader range of applications.

## 5 Educational Impact and Future Directions

The research challenges outlined in this paper have provided substantial research opportunities for students at the undergraduate, M.S., and Ph.D. levels, and their work on these problems has enabled them to put into practice the principles and techniques learned in the courses we have described. In addition to the extensive research that has gone into designing and building the integrated simulation code, the integrated code has itself become a research tool for use by students in studying complex multicomponent physical systems. The resulting educational opportunities include the following:

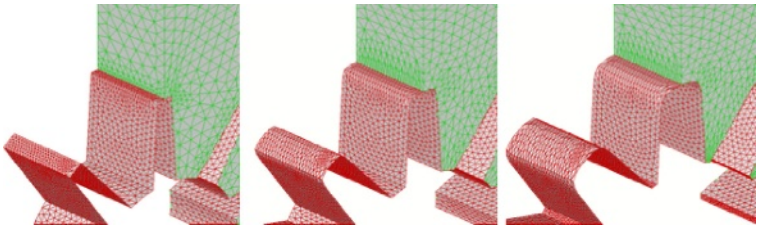

**Fig. 5.** Initial burn of star slice exhibits rapid expansion at slots and contraction at fins. Three subfigures correspond to 0%, 6%, and 12% burns, respectively

- Learning about interactions between different kinds of physical models using the coupled code with small problems,
- Adapting independently developed codes to include the coupling interfaces necessary to interact with other physics or service modules,
- Exploring new coupling schemes using high-level, domain-specific constructs.

Our interdisciplinary team has made substantial achievements in advancing the state-of-the-art in multiphysics simulations, but numerous challenges remain, among which are the following:

- Distributed algorithms and data structures for parallel mesh repair and adaptivity,
- Parallel mesh adaptation for crack propagation in three dimensions,
- Data transfer at sliding and adaptive interfaces.

These and other new research opportunities will require close collaboration between computer and engineering scientists to devise effective and practical methodologies. The implementation of these capabilities can be decoupled, however, owing to our flexible integration framework, and can be accomplished relatively quickly by leveraging the existing abstractions and service utilities of our infrastructure. Through this empowerment of individual students as well as interdisciplinary teams, we believe that our integration framework can play a significant role in training the next generation computational scientists and engineers.

## References

1. L. Freitag, T. Leurent, P. Knupp, and D. Melander. MESQUITE design: Issues in the development of a mesh quality improvement toolkit. In *8th Intl. Conf. on Numer. Grid Gener. in Comput. Field Simu.*, pages 159–168, 2002.
2. M. T. Heath and W. A. Dick. Virtual prototyping of solid propellant rockets. *Computing in Science & Engineering*, 2:21–32, 2000.
3. X. Jiao, M. T. Campbell, and M. T. Heath. Roccom: An object-oriented, data-centric software integration framework for multiphysics simulations. In *17th Ann. ACM Int. Conf. on Supercomputing*, pages 358–368, 2003.
4. X. Jiao and M. T. Heath. Common-refinement based data transfer between non-matching meshes in multiphysics simulations. *Int. J. Numer. Meth. Engrg.*, 61:2401–2427, 2004.
5. X. Jiao and M. T. Heath. Overlaying surface meshes, part I: Algorithms. *Int. J. Comput. Geom. Appl.*, 14:379–402, 2004.
6. X. Jiao, M. T. Heath, and O. S. Lawlor. Face-offsetting methods for entropy-satisfying Lagrangian surface propagation. In preparation, 2004.
7. S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer, 2003.
8. J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.