3.5 Computer Science

Group Leaders: Michael Heath and Laxmikant Kale

Faculty: Eric de Sturler, Herbert Edelsbrunner (Duke University), Michael Heath, Laxmikant (Sanjay) Kale, David Padua, Daniel Reed, Paul Saylor, Shang-Hua Teng, and Marianne Winslett

Research Scientists/Programmers: Milind Bhandarkar, Michael Campbell, Robert Fiedler, Xiangmin (Jim) Jiao, Orion Lawlor, John Norris, and Kent Seamons

Post-Doctoral Associates: Damrong Guoy, Joerg Liesen, and Alla Sheffer

Graduate Research Assistants: Gheorge Almasi, Yelda Aydin, William Cochran, Damrong Guoy, Xiangmin (Jim) Jiao, Jonghyun Lee, Xiaosong Ma, Greg Mackey, Ali Pinar, Joseph Pepper, Neelam Saboo, Alper Ungor, Terry Wilmarth, Shengke Yu, Zhe (Robert) Zhang, Jie Zheng, and Jiajing Zhu

Consultant: Timothy Baker (Princeton University)

Overview

Two teams, Computational Mathematics and Geometry, and Computational Environment, address research in Computer Science. The target of our Computational Environment team is to provide the broad computational infrastructure necessary to support complex, large-scale simulations in general, and for rocket simulation in particular. Areas of research include parallel programming environments, compiler optimization and parallelization, parallel performance analysis and tools, parallel input/output, and visualization. Work in Computational Mathematics and Geometry focuses on the areas of linear solvers, mesh generation and adaptation, and interpolation between diverse discretizations.

Computational Environment

Parallel Programming Environment (Bhandarkar, Kale, Lawlor)

Our focus in the past year has been to leverage our existing parallel programming infrastructure to simplify development of adaptive parallel applications. Our parallel programming system consists of the *Charm++* runtime library and load balancing framework, and its multi-threaded extensions. In the past year, we have built upon our preliminary implementation of Adaptive MPI (AMPI), which allows MPI applications to adapt to varying availability of resources. We have added the capability of coupling multiple MPI applications to AMPI using remote communicators. Using this extension, multiple MPI modules can co-exist within a single parallel application, communicating using familiar MPI calls. Each module runs in its own set of parallel threaded objects, thus allowing the runtime system to load balance them independently. We re-implemented the GEN1 codes using this architecture, where *Rocflo* and *Rocsolid* are treated as completely separate applications that communicate using MPI calls.

A flexible checkpoint/restart facility was added to AMPI. It utilizes application-provided information regarding migrating state of the virtual processors, and allows any AMPI application to be checkpointed and restarted on a (possibly) different number of processors. Conversion of existing MPI application to AMPI is a fairly mechanical (and therefore tedious) task. We have developed AMPIzer, a source-to-source translator for *automatic conversion* of MPI/Fortran programs to AMPI. AMPIzer utilizes the front-end of the Polaris compiler suite developed at Illinois by Prof. David Padua and colleagues. AMPIzer performs privatization of global variables in Fortran programs, so that multiple instances of code can co-exist within the same address space using AMPI threads. We have demonstrated AMPIzer on several benchmark codes and a real application code, *Rocfire*.

Many engineering codes, such as those that employ the Finite Element Method (FEM), have similar parallel control structures. We have developed an adaptive FEM framework that abstracts these control structures. The application developer merely specifies the sequential processing to be done on finite elements, and the remaining code is supplied by the framework, greatly accelerating development of parallel applications. The FEM framework focuses on dynamic and irregular applications, and has been found useful in implementing several applications in CSAR as well as the Center for Process Simulation and Development (CPSD).

We are currently working on an application framework for multi-block CFD codes. This framework is motivated by the success of our adaptive FEM framework, which has been shown to reduce application development time significantly. The multi-block framework will be powerful enough to support complex fluid dynamics applications such as *Rocflo*.

Control and data interoperability among application frameworks is of critical importance for complex physical simulations. In the future, our focus will be on developing new paradigms for application integration. Research is underway for providing a component architecture for parallel application frameworks based on message-driven, multi-domain decomposition. An innovative interface model for this component architecture has been proposed, and is currently being implemented. Our efforts will be complementary to the software integration framework efforts of other researchers in CSAR. In particular, we will provide a substrate upon which the component architecture of GEN2 integrated code, *Roccom*, can be developed.

Particle-based (mesh-less) physical simulations are gaining in importance. Our experience in molecular dynamics applications (where such simulations are often employed) provides us with the opportunity to build on existing abstractions to evolve into a particle-based simulation framework. We plan to develop this framework to add to our existing infrastructure of application frameworks for physical simulations.

Software Integration Framework (Bhandarkar, Campbell, de Sturler, Fiedler, Jiao, Kale, Lawlor)

We are developing a software environment, called *Swift*, for coupling stand-alone parallel applications. This environment allows multiple applications to interact in jointly simulating complex multi-physics applications, such as the burning of a solid propellant rocket. The main idea behind our software environment is that applications require only minimal changes to use it, hence multiple stand-alone applications can be coupled in a matter of days or weeks rather than months or years. The main part of our software environment for coupling is finished, although the various parts are still evolving. We spent a significant amount of time testing and profiling (timing) the GEN1 codes to ensure that no performance degradation occurs. A software component based standard for integrating applications and the framework has been developed (*Roccom*), and APIs for coupling have been implemented in C++ (mainly framework side) and Fortran90 (mainly application side). *Roccom* allows the framework to be used with various parallel systems, including *Charm++, Autopilot*, or basic MPI based implementations.

As part of the software framework effort, we have developed a new software component, *Roccom*, which provides a mechanism for data exchange and function invocation between *Roc** components. With *Roccom*, each *Roc** component manages its own data and registers with *Roccom* the addresses of its data using publicized names. A module references data of other modules by these names and accesses them through *Roccom*. Invoking a function in another module is also done through *Roccom* using publicized names, and the actual function that is invoked is determined at runtime and can be performed locally or remotely, transparent to the caller. Only minimal changes are required to an existing physics code to fit into this framework. *Roccom* has promoted modularity, enabled plug-and-play of components, and improved productivity in integrating *Roc** components into GEN2.

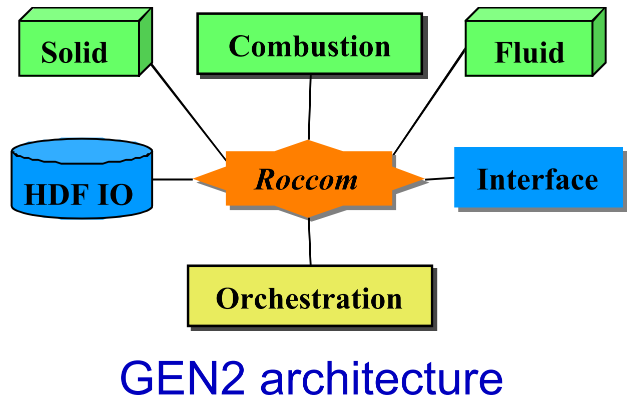

Fig. 3.4.1: CSAR *Roc** architecture employs *Roccom* for data exchange between modules.

Further development of *Roccom* in the near future will focus on the integration of other software framework efforts, development of add-on components, and support for dynamically changing environments. Our two other major framework efforts, *Autopilot* and *Charm++*, both provide *Roccom* implementations, which can be substituted for the basic *Roccom* implementation in GEN2, allowing integration of these tools into GEN2 with almost no changes to other components. One of the planned add-on components of *Roccom* is *Rocman*, which performs computational manipulation of data registered to *Roccom* and encapsulates the interface physics. This and other add-on components will further boost the modularity and code reuse in GEN2. In order to support the dynamic behavior of GEN2 and GEN3 introduced by adaptive remeshing, *Roccom* is planned to provide dynamic load balancing through *Charm++* and on-the-fly module substitution to enhance efficiency.

During the next year we will port the new GEN2 code to the *Swift* software environment. The plan is that from then on all new developments will be integrated immediately with the environment. We will also start implementing more general orchestration and steering capabilities in the software environment. In addition, several tools are being developed to assist developers in adapting their applications for integration with the software environment.

Parallel Input/Output (Lee, Ma, Seamons, Winslett)

We designed an aggressive buffering scheme called "greedy buffering" and implemented it in *Rocpanda*, our high-performance parallel I/O library. Experiments with *Rocpanda*, GEN1, and GEN2 on the SP, Origin 2000, and Linux cluster showed that greedy buffering can greatly reduce the apparent I/O cost for simulation applications, through better use of idle memory and idle time on processors used for I/O.

We proposed performance models for different data migration strategies and showed how to use the models to predict and tune migration performance for parallel scientific simulations such as GEN1. We also showed how data compression can be used to reduce the cost of migration.

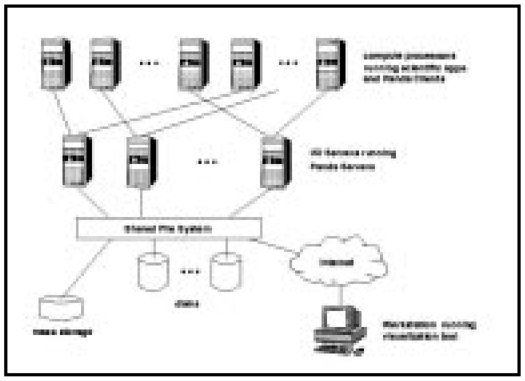

Fig. 3.4.2: *Rocpanda* architecture sets aside a few processors for communication.

We adopted a new general search method called DLM that speeds up the *Panda* I/O optimizer by up to a factor of 50. We began work on developing a platform-independent performance meta-model, which could generate a performance model for parallel I/O on a particular platform automatically and efficiently, thereby helping to make a portable parallel I/O optimizer.

In the next year, we will push our new parallel I/O greedy buffering scheme further by exploiting the full storage hierarchy for buffering, from client-side memory buffers to serverside disks and even remote storage. We will also work on the automatic selection of the best data migration strategy in a given computing environment, and explore the integration of data compression with buffering to reduce data migration costs further. We also plan to finish our work on creating a portable and modular parallel I/O optimization approach. Finally, we will continue to meet new GEN2 I/O needs as they arise.

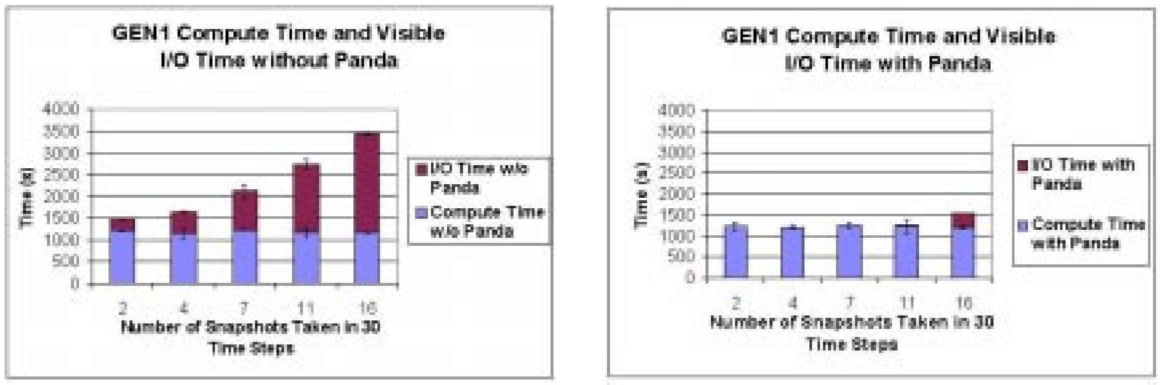

Fig. 3.4.3: *Rocpanda* significantly reduces the I/O time in multiprocessor runs.

Scientific Visualization (Fiedler, Norris)

Rocketeer, a 3-D scientific visualization tool developed at CSAR, has continued to advance in power and sophistication. Existing visualization techniques have been expanded, new techniques have been added, and a wide range of speed optimizations have been implemented. The "glyphs" technique has been extended for visualizing vector fields. A new "mesh" technique allows the user to draw all or part of a 3-D mesh. In addition, support for 2-D surface meshes (2-D elements in 3-D space) has been enhanced. These advances are also present in *Voyager*, a new MPI parallel command-line visualization tool that creates images from a series of output dumps concurrently.

The client/server version of *Rocketeer*, called *Apollo/Houston*, is now being tested. The current alpha version has nearly all of the functionality of *Rocketeer*. *Houston*, an MPI parallel server, handles reading the data and the generation of isosurfaces, glyphs, etc. The resulting polygonal data are sent over the network to *Apollo*, which renders the final, combined image at the desktop, exploiting hardware graphics acceleration.

Development of these visualization tools will continue during the coming year. Support for large data sets and various data file formats will be added.

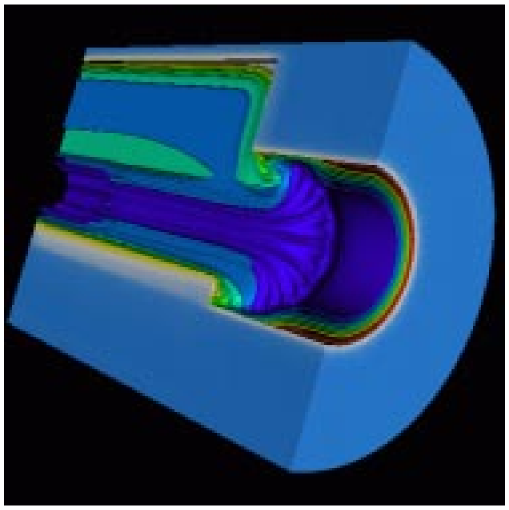
Fig. 3.4.4: *Rocketeer* image showing temperature isosurfaces in head end of RSRM shortly after ignition.

Computational Mathematics and Geometry

Mesh Generation, Adaptation, and Repair (de Sturler, Guoy, Sheffer)

We have developed two algorithms as building blocks for mesh repair problems for unstructured tetrahedral meshes. The first algorithm, called *sink insertion*, applies point insertion to refine the mesh selectively. The second algorithm, called *sliver exudation*, applies local reconnection controlled by weight assignments. Sink insertion is a variant of Delaunay refinement. Given an input Delaunay mesh, we define a continuous scalar field that is differentiable almost everywhere. The local extrema of the scalar field correspond to points that are suitable for insertion into the mesh. It turns out that these points are circumcenters of tetrahedra such that the circumcenter lies inside the tetrahedron. This process is capable of eliminating every type of poor quality tetrahedra except slivers. Sliver exudation assigns weights to vertices in order to control the connectivity between vertices. The process replaces Euclid-

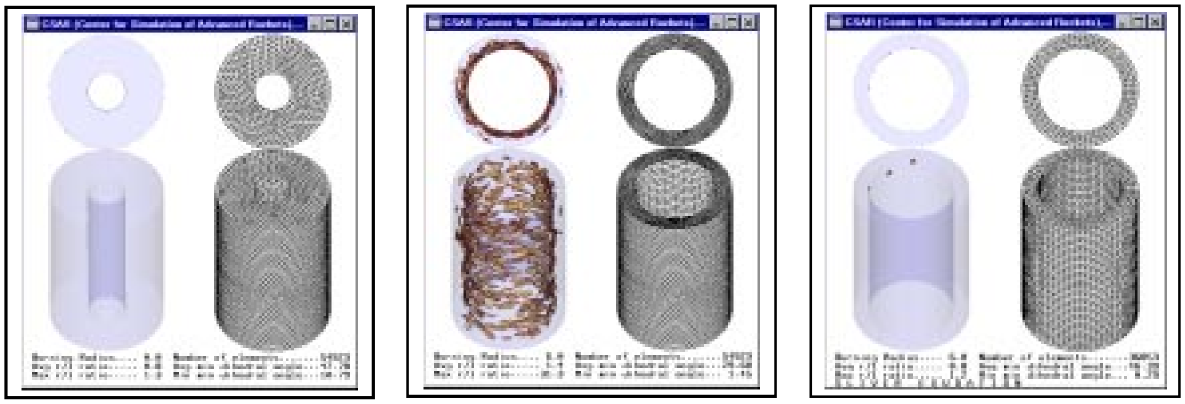
Fig. 3.4.5: Mesh repair is critical to solid rocket combustion simulations. Shown here are initial mesh prior to any combustion, “bad” elements and mesh after 60% of propellant has burned away, and repaired mesh.

ean distance by weighted distance. The Delaunay triangulations become weighted Delaunay triangulations, which is a well-studied construction for the past decade. We performed computational experiments to demonstrate the effectiveness of the two algorithms. Future work on mesh repair will consist of repairing the surface mesh before the volume mesh, derefinement algorithms, and parallelization.

During the coming year, we plan to investigate the dynamic meshing of evolving 3-D domains. Because the domains can shrink and expand in complicated patterns, good quality meshes need to adapt accordingly. A major challenge is that the evolution of the domains is not known in advance, and therefore the meshing routines must be robust and fully automatic.

We developed a new algorithm for the parameterization of tessellated surfaces that significantly advanced the state of the art in functionality and robustness. Our algorithm can treat more complicated surfaces and compute parameterizations with less deformation than alternative methods, and it guarantees correct results. The algorithm computes a mapping from a tessellated three-dimensional surface to the plane. This problem arises in the generation of three-dimensional meshes, but also in a variety of other applications such as texture mapping in graphics. The two most important innovations in our algorithm are basing the parameterization entirely on the angles in the triangles (or other polygons in the tessellated surface) and reformulating the problem as a constrained optimization problem. This project is also sponsored by Sandia National Labs, where we collaborate with Steve Owen.

Meshes coming from CAD packages are intended mainly to describe geometry and are not useful for finite element analysis. Hence the objects often need to be remeshed. Figure 3.4.6 is an example for a model problem (Isis statue). In fact, this model is harder than most real-world problems. In the figure (a) shows the original surface mesh, (b) shows a flattened mesh with minimal angular deformation (linear deformation is handled through a spacing function), (c) shows a coarser mesh where the triangles have much better aspect ratio, and (d) shows the new mesh mapped back to the surface of the object.

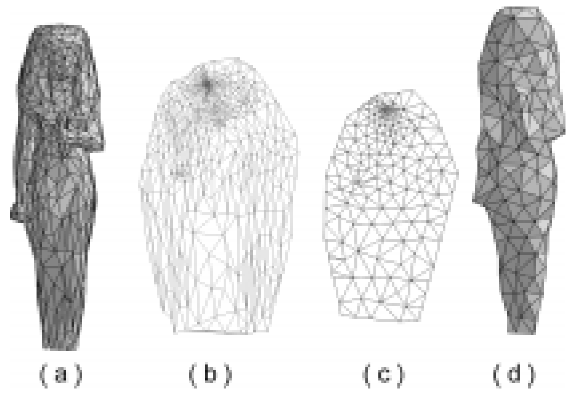

Fig. 3.4.6: Original mesh and remeshing strategy for FE analysis.

Data Transfer Across Interface Surfaces (Heath, Jiao)

During the past year, the main focuses of *Rocface* have been on developing efficient and robust algorithms for constructing a common refinement of two surface meshes, and on investigating accurate and conservative data transfer schemes between nonmatching meshes. The nonmatching meshes handled by *Rocface* can have different combinatorial structures, geometric realizations, and partitionings. A *common refinement* provides a unique reference mesh between these nonmatching meshes, and allows efficient query of a unique nearby corresponding point on one surface for every point on the other. We have developed an efficient algorithm with linear complexity for constructing a common refinement. The primitives of the algorithm involve nonlinear equations, which are solved approximately and efficiently using floating-point arithmetic. To address robustness with imprecise computations, we defined a set of consistency rules and an intersection principle for detecting and resolving the inconsistencies caused by numerical errors. Techniques for detecting and matching special geometric features in the meshes (such as sharp edges and corners) were also developed to further enhance robustness and improve the quality of the common refinement.

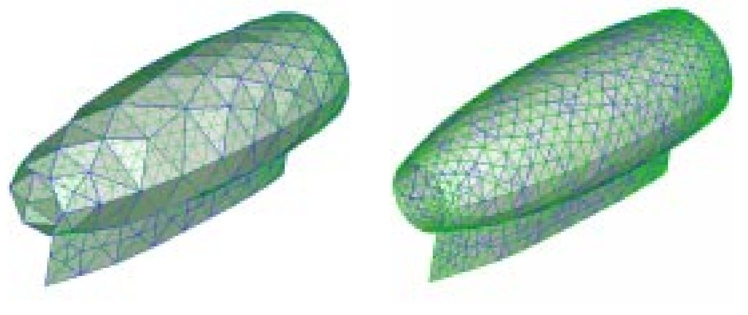
Fig. 3.4.7: Engine nacelles illustrate realizations of common refinement of coarse (left) and fine (right) meshes.

A common refinement of two meshes provides a very efficient and convenient data structure for implementing new *data transfer* algorithms between meshes. In particular, we have developed a class of least squares data transfer schemes that by construction minimize the error in the Euclidean or energy norm while also enforcing global conservation. These schemes require integration over the intersections of the cells of the input meshes, which are enabled by the common refinement. Our algorithms can transfer both node- and element-centered data, and comparison with traditional methods shows substantial advantages for the new methods. Parallelization of these algorithms was also made easy and efficient by the common refinement, which contains all necessary communication information between the meshes. The algorithms for both common refinement and data transfer have been implemented and integrated into the GEN2 rocket simulation code.

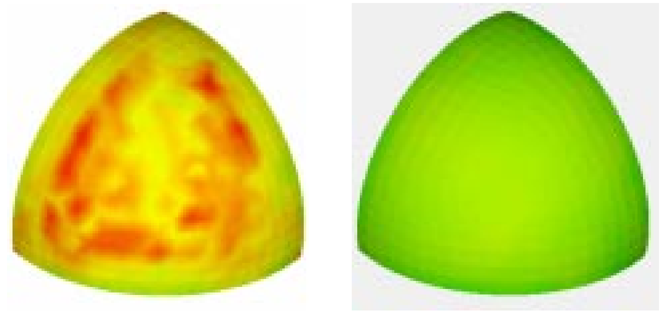
Fig. 3.4.8: Accuracy of data transfer for burning cavity problem. Accumulated error for conventional method (left) exceeds 20%, but for new method (right) is less than 1%.

In the coming year, we plan to focus on interface propagation in *Rocface*. Interface propagation is concerned with moving the interface between two meshes given the speed at points on the interface. We have developed a new approach for interface propagation that combines the advantages of both level-set methods and marker particle methods. In order to integrate our method into the rocket simulation code, we must overcome the additional complexity of moving more than one interface mesh synchronously, and the common refinement of the meshes is expected to play a significant role in this setting. In addition, we also plan to extend the common refinement and data transfer schemes to volume meshes so that data can be mapped accurately and conservatively after adaptive mesh refinement, or coarsening, or remeshing.

Solvers and Preconditioners for Linear and Nonlinear Problems (de Sturler, Liesen, Saylor)

Much of our work on linear solvers focused on Minimal Residual (MR) and Conjugate Gradient (CG) methods, which are among the most widely used Krylov subspace methods for solving non-hermitian linear systems. Many popular implementations of MR methods are mathematically equivalent, but they employ different bases and can differ numerically. Using new results from the theory of least squares, we showed that in some well-known implementations an inappropriate choice of basis leads to numerical instability unrelated to the given data. These implementations may perform poorly even for very well-conditioned problems.

We also studied B-normal matrices, which are important in the context of CG methods. In addition to new characterizations of B-normality and ordinary normality, we gave a new, elementary proof of the fundamental theorem of Faber and Manteuffel on the existence of short-term recurrence CG methods.

We extended the theory of a special class of preconditioners for symmetric indefinite linear systems. Such problems arise in a method for (tessellated) surface parameterization that we developed. These preconditioners are important for many other applications in optimization, simulation of conservative physical processes, and meshing. We also began evaluating the optimally truncated GMRES method for problems with multiple successive right-hand sides in collaboration with Philippe Geubelle's group. Such problems occur, for example, in fluid-structure interactions with small deformations (vibrations) in the structure. In addition, we worked on methods for solving nonlinear and optimization problems with very ill-conditioned Jacobians. Preliminary experiments showed improvement over other methods.

Work continued on solution methods for radiation transport in collaboration with Doug Swesty of SUNY Stony Brook and Dennis Smolarski of Santa Clara University. Favorable results with sparse approximate inverse preconditioners encourage further development. Specific investigations of properties of subblocks of transport matrices related to efficient methods for matrix-vector multiplication. If it happens these are low rank, then matrix multiplication for a k by l subblock becomes as efficient as for a sparse matrix. Investigations in collaboration with Gene Golub of Stanford and Gerry Minerbo of Schlumberger Technology on using extended matrices to compute scattering cross sections yielded a viable method.

Our work on MR methods will next examine the question how the interplay of eigenvalue-eigenvector structure of the system matrix and the given right-hand side influence the convergence behavior. The new results on the existence of CG methods pave the way for the discovery of other short-term recurrence methods and further analysis of existing ones. After successful tests on small problems, large scale testing and additional mathematical analysis of the new preconditioners for indefinite linear systems are underway.

Other future work involves the integration of iterative solvers with the structures simulation code, evaluating the usefulness of a method for linear solvers for systems with multiple successive right-hand sides (requires fewer and fewer iterations per right-hand side), improving preconditioners for symmetric indefinite linear systems, and assessing the possibilities to use comparable techniques for nonsymmetric systems.