3.6 System Integration

System Integration Coordinators: Amit Acharya, Eric de Sturler, Robert Fiedler, Michael Heath, Jay Hoeflinger, Thomas Jackson, Fady Najjar, and Dennis Parsons

Faculty: Eric de Sturler, Michael Heath, Laxmikant (Sanjay) Kale, David Padua, and Dennis Parsons

Technical Program Manager: Robert Fiedler

Research Scientists/Programmers: Amit Acharya, Prasad Alavilli, Jay Hoeflinger, Thomas Jackson, Fady Najjar, Alireza Namazifard, John Norris, and Danesh Tafti

Post-Doctoral Associate: Alla Sheffer and Ertugrul Taciroglu

Graduate Research Assistants: Jason Hales and Xiangmin Jiao

Overview

Detailed simulation of solid propellant rockets requires the development of computer codes that implement mathematical models of each rocket component and the interactions between them. These components include the case, propellant, combustion layer, core gas, and nozzle. System integration involves control of component code execution, exchange of information between components ("boundary data"), and temporal coupling of components ("time stepping").

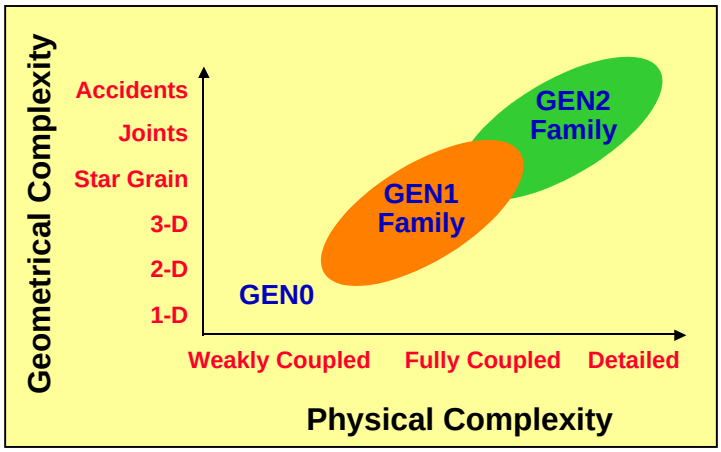

Fig. 3.6.1. CSAR follows a "staged" approach to system integration.

As a major step toward the full GEN1 system code (Fig. 3.6.1) , we developed a simple, but fully coupled, three-dimensional rocket simulation. The main purpose of this early version was to gain experience in system integration by coupling together modified versions of existing, locally developed, three dimensional fluid dynamics and structural dynamics codes. The modules have the following features:

- Combustion: mass is injected at a pressure-dependent rate and fixed temperature normal to the burning propellant surface. For short burn times, we may neglect regression of the propellant, but not its deformation driven by the pressure of the fluid. Appropriate jump conditions are used to conserve linear momentum across the infinitesimally thin combustion interface.
- Fluids: the fully compressible Euler equations are solved throughout the core flow region using *ROCFLO*, an explicit finite volume code developed by Alavilli. *ROCFLO* features a moving body-fitted coordinate system (ALE formulation) and a second-order accurate upwind TVD scheme.

- Structures: The cylindrical grain propellant is taken to be linearly elastic and compressible, while the case is taken to be rigid. *ROCSOLID*, a multigrid finite element code developed by Namazifard and Parsons, is employed to solve the equation of motion. Tractions at the propellant surface are computed using the fluid pressure, fluid velocity, and jump conditions. *ROCSOLID* takes implicit time steps, which are typically ten times larger than the time steps taken by *ROCFLO*.

ROCFLO, *ROCSOLID*, the main program, and the initial interface code were all written in Fortran90. Interpolation between the fluids and structures grids was simplified in this early version of GEN1 by constraining the two grids to coincide at the combustion interface.

The Structures and Fluids modules are temporally coupled through a "predictor-corrector" type algorithm in which time steps are taken separately in the Fluids and Structures modules, iterating back and forth between modules until the convergence criteria are satisfied.

The serial version of GEN1 (previously known as "GENH," for GENHalf) was completed in October 1998. The parallel version of the first implementation, completed in December 1998, used MPI for message passing in the fluids, structures, and interface codes.

The full GEN1 system code includes more physical and geometrical complexity than the earlier version, which allows meaningful comparison with the Space Shuttle RSRM (Fig. 3.6.2). *ROCFLO* now solves the Navier-Stokes equations and can handle the star grain region at the head end. The head end pressure just after steady burn is achieved computed by *ROCFLO* is within 10 percent of the actual value for the RSRM. Alavilli has added a geometry-based algorithm to *ROCFLO* for tracking the regressing propellant surface, and Xiaosong Ma has incorporated *Panda* for parallel I/O.

The GEN1 system code includes a linear elastic case. An enhanced version of *ROCSOLID* in which the regressing propellant surface is followed using an ALE formulation is currently being tested against analytical solutions. We are also developing a viscoelastic model for the propellant and extending the code to handle large deformations. *Panda* will soon be added to *ROCSOLID* for parallel I/O.

In GEN1, the fluids and structures meshes are not required to match at the interface. Jiao has developed an efficient algorithm to associate points in the fluids mesh with elements in the Structures mesh. This Interface code also performs interpolation in a manner that guarantees conservation of mass and

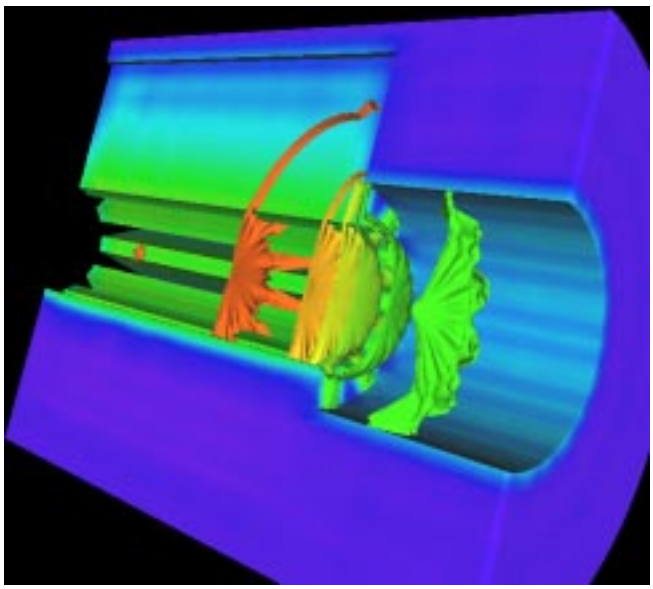

Fig. 3.6.2. Full 3-D simulation of star grain in Space Shuttle RSRM showing stress in propellant and gas pressure isosurfaces in slots and core region. Executed on 256-processor SGI Origin2000, visualized with *Rocketeer*.

linear momentum across the surface (see Farhat, Lesoinne, and Maman, 1995). An MPI parallel version of this C++ code is nearing completion.

The GEN1 component codes will be modified to utilize the software framework under development by Kale, de Sturler, and Hoeflinger. This framework performs dynamic load balancing automatically, which becomes important as the propellant burns away and the computational volumes occupied by the Fluids and Structures modules change. The framework also simplifies communication between components through its data port and mesh matching objects.

Scientific visualization and animation of our simulations is facilitated by *Rocketeer*, developed by Norris and Fiedler. *Rocketeer* is based on the Visualization Toolkit, which is written in C++ and uses OpenGL to take advantage of graphics hardware acceleration. The User's Guide is available at http://www.csar.uiuc.edu/F_software/rocketeer/Rocketeer_Users_Guide.htm.

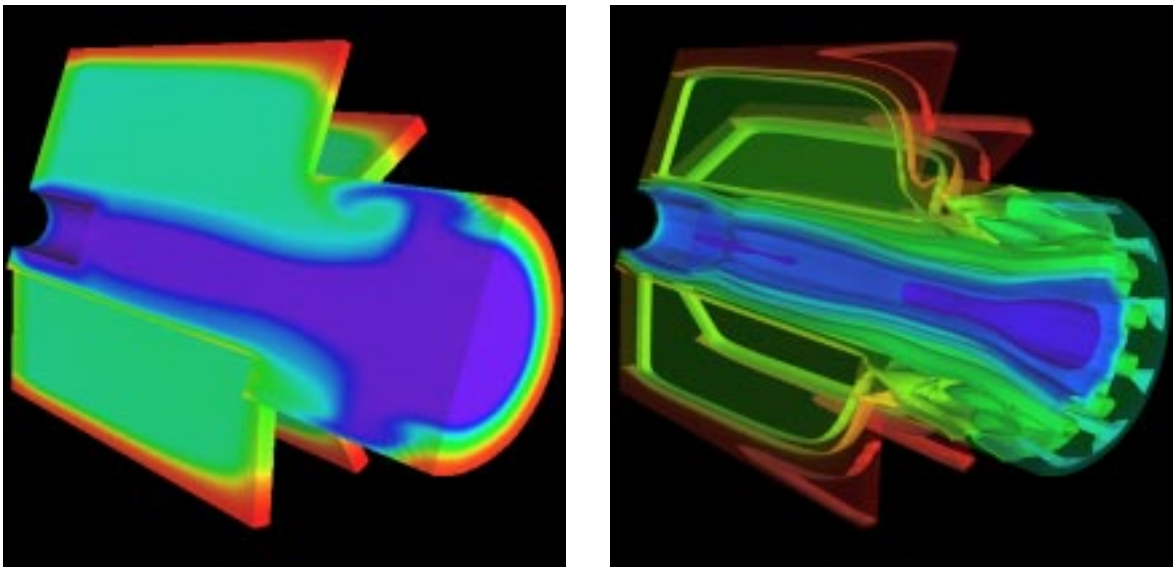

Figure 3.6.3. Gas temperature computed by *ROCFLO* in star grain region of Space Shuttle RSRM near onset of steady burning, visualized by *Rocketeer*. Values range from 3364 K (magenta) to 3392 K (red). Temperature is represented as tint on interior surface of propellant (left image) and as series of translucent colored isosurfaces in interior at slightly later time (right image). Rocket has been cut in half along lateral axis to improve visibility.

Key features of *Rocketeer* include:

- Support for unstructured, multiblock, and adaptive grids
- Displays grids, values on exterior surfaces, slices, isosurfaces, multiple data sets, color scales, etc.
- Adjustable transparency aids in viewing 3-D volumes
- Intuitive GUI automates repetitive tasks such as merging data files and generating animation frames
- Smart reader for NCSA's HDF format interprets content of data files

The initial release of *Rocketeer* runs on Windows. A port to wxWindows is well underway, which will allow it to run on UNIX and other platforms as well. Our planned enhancements include adopting a client/server model so that most compute intensive operations can be performed (in parallel) on a remote supercomputer while interactive control and rendering are performed locally on a graphics workstation.