3.6 System Integration

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

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

Technical Program Manager: Robert Fiedler (SITeam Chair)

Research Scientists/Programmers: Amit Acharya, Prasad Alavilli, Michael Campbell, Andreas Haselbacher, Jay Hoeflinger, Thomas Jackson, Fady Najjar, Alireza Namazifard, John Norris, Danesh Tafti, and KC Tang

Post-Doctoral Associates: Ramesh Balakrishnan, Biing-Horng Liou, Changyu Hwang, Alla Sheffer, and Ertugrul Taciroglu

Graduate Research Assistants: Jason Hales, Xiangmin Jiao, and Xiaosong Ma

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").

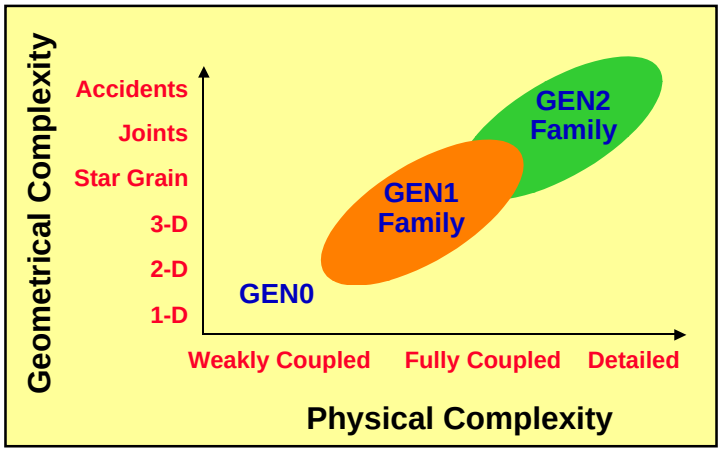

Fig. 3.6.1: CSAR follows "staged" approach to system integration.

As our initial step toward the full GEN1 system code (Figure 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 assumed 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*, 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 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 (Figure 3.6.2). *Rocflo* now solves the Navier-Stokes equations and can handle the star grain region at the head end.

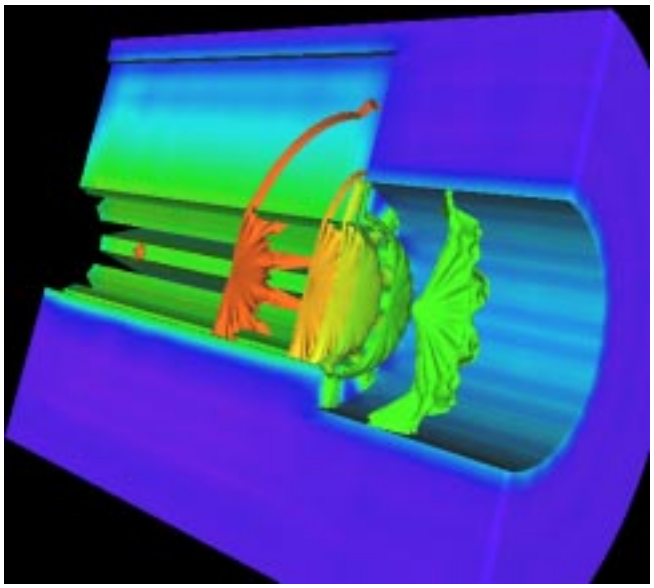

Fig. 3.6.2: Full 3-D simulation of star grain in Space Shuttle RSRM showing stress in propellant and gas pressure isosurfaces in slots and core region. Executed on 256-processor SGI Origin2000, visualized with *Rocketeer*.

The GEN1 integrated 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* has been added this year 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. The interface code also performs interpolation in a manner that guarantees conservation of mass and linear momentum across the surface (see Farhat, Lesoinne, and Maman, 1995). An MPI parallel version of this C++ code has been completed this year.

The GEN1 component codes have been partially modified to utilize the evolving software framework being developed by Kale, de Sturler, and Hoeflinger. This framework per-

forms 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.

Visualization

Rocketeer, developed by Norris and Fiedler facilitates scientific visualization and animation of our simulations. *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. *Rocketeer* is a 3-D scientific visualization tool. It was rewritten in Y3 using *wxWindows* for cross-platform portability to run on Sun and Linux systems. Earlier versions run on PC and Mac systems.

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

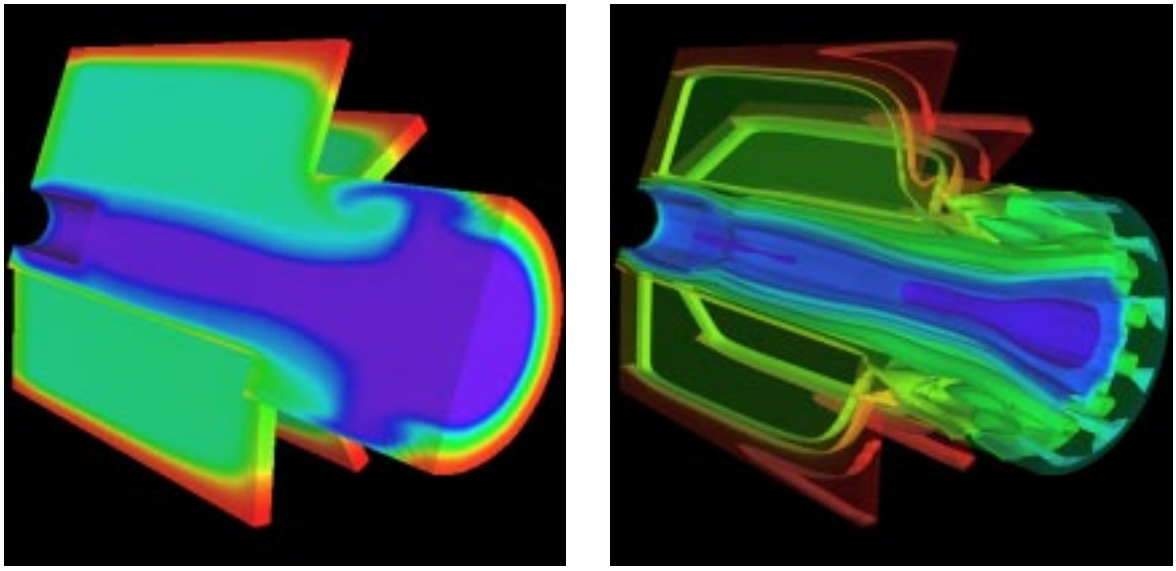

Figure 3.6.3: Gas temperature computed by *Rocflo* in star grain region of Space Shuttle RSRM near onset of steady burning visualized by *Rocketeer*. Values range from 3364 K (magenta) to 3392 K (red). Temperature is represented as tint on interior surface of propellant (left image) and as series of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

An enhanced version of *Rocketeer* (1.1) enables drawing spheres at a set of points, useful for showing the packing of particles in solid propellants and for showing aluminum droplets in the flow. Another recent enhancement is the ability to plot structural mechanics data in undeformed coordinates, deformed coordinates, or both on the same image. The displacements can be exaggerated by a user-selectable factor. Support has been added for discontinuous Galerkin finite element meshes and 2-D triangular meshes, as well. New features in Y3 include:

- Ability to select vector data to use to distort the grid, as well as the ability to select a scale factor.
- New visualization technique, *spheres*, which renders spheres at a random subset of the dataset's points. The spheres' color and/or radius are controlled by their corresponding scalar values.
- Ability to specify external geometry files, which can save a considerable amount of disk space when dealing with several dumps of large datasets when the grid's geometry never (or rarely) changes.
- Support for a new grid type, *discontinuous*, which is an unstructured grid where the nodes of each element have unique ids, even if they coincide.

Fiedler worked with Ma to add *Panda* for parallel I/O to *Rocsolid*. This new version is being combined with the *Panda*-ized version of *Rocflo* to complete a version of the GEN1 code with parallel I/O (HDF format) and background file transfer capabilities.