3.6 System Integration and Simulations

Team Coordinators: Mark Brandyberry and Robert Fiedler

Research Engineer: Courtlan McLay

Research Programmer: Johnny Norris

Undergraduate Students: Joseph Tortorelli and Rebecca Wingate

Overview

This year's efforts were devoted to extending the capabilities of the *Rocstar* version 3 rocket simulation package, verifying and validating the coupled code, and performing several large-scale simulations.

Rocstar Development

A diagram of the *Rocstar* 3 architecture is shown in Figure 3.6.1. A brief description of the specific modules that perform the functions written in each box is given below. Several new modules were fully integrated into *Rocstar* and several existing modules were significantly enhanced in order for us to apply our code to new problems of current scientific interest. The basic numerical methods and capabilities are summarized here. Many of these modules are described in more detail elsewhere in this document.

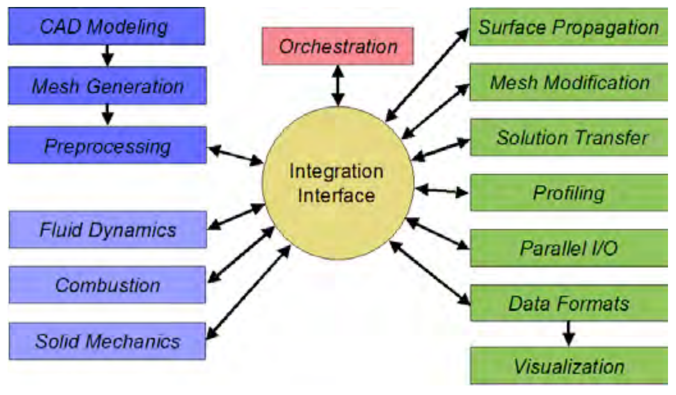

Fig. 3.6.1: *Rocstar* 3 architecture

Input Dataset Preparation

On the left-hand side of Figure 3.6.1, the problem-definition tools and the physics solvers are represented by blue boxes (lighter blue for the solvers). The selection of CAD packages is up to the user, as long as the package can output the geometrical information needed by the mesh generator(s). To some degree, the mesh generator may also be chosen by the user, although the physics application developers have written preprocessors that require mesh and boundary condition information in a very specific format. Our intention is to provide reader routines that support a number of commonly used mesh generators along with the preprocessors for the physics applications, which will allow the user to select any supported meshing tool.

Once the meshes and boundary condition information are written in a supported format, the physics application preprocessors can be run either by hand, or (preferably) with the aid of the *Rocprep* input data set preparation tool (completed this year). The preprocessors create complete input data sets partitioned for parallel execution for each physics application. This year, all preprocessors were rewritten to make use of the *Rocin/Rocout* I/O modules so that initial data sets are nearly identical in form to restart data sets, facilitating remeshing. Initial meshes and boundary conditions can now be visualized in the same manner as normal output dumps.

Physics Solvers

The three light blue boxes on the lower left in Fig. 3.6.1 represent the various general-purpose physics solvers that are available for use with *Rocstar*. The existing fluid dynamics packages are *Rocflu* and *Rocflo*. The basic algorithms in these codes were pioneered by Jameson. *Rocflu* operates on unstructured tetrahedral or mixed tetrahedral/hexahedral/pyramid/prism mesh cells to handle complex geometries. An advantage of mixed meshes is the ability to use hexahedral cells to provide high spatial resolution in boundary layers near physical surfaces. The fluid equations are formulated on moving meshes (Arbitrary Lagrangian Eulerian, or ALE scheme) to handle geometrical changes such as propellant burning and deformation. This finite-volume code employs a new high order WENO-like approach, as well as the HLLC scheme to handle strong transient such as igniter flows. Time integration is accomplished via either the 3rd or 4th order explicit multistage Runge-Kutta time stepping algorithm, or by a newly implemented Newton-Krylov-based implicit scheme that utilizes the PETSc library. *Rocflo* uses either the central scheme or an upwind scheme involving Roe flux splitting on multi-block structured meshes. In addition to explicit Runge-Kutta, *Rocflo* can use a newly implemented dual time stepping algorithm to take time steps longer than the Courant (CFL) limit. Both fluid solvers can now include turbulence (*Rocturb*), Lagrangian superparticles (*Rocpart*), smoke (*Rocsmoke*; equilibrium-Eulerian method), chemical reactions (*Rocspecies*), and radiation (*Rocrad*; diffusion approximation).

The rate of propellant deflagration is computed by one of three combustion modules. These physical models are one-dimensional (normal to the surface) in formulation, but are applied independently at each cell face on the burning propellant surface, making them effectively 3-D. The simplest model, *RocburnAPN*, adopts the well-known steady burn rate model in which the regression speed is proportional to the local gas pressure raised to the power "*n*." Two dynamic burn rate models may also be selected. Both solve a 1-D time-dependent heat conduction equation for the temperature profile in order to capture ignition transients. One of the dynamic models (*RocburnZN*) is based on the Zeldovich-Novozhilov approach, while the other (*RocburnPY*) uses a simpler pyrolysis law. *RocburnPY* can also compute the heating of the propellant surface by hot igniter gases prior to burning, as well as ignition once the critical temperature is exceeded. New this year: a heat-flux look-up table computed by *Rocfire*, the detailed 3-D propellant combustion simulation code developed at CSAR, can be used by *RocburnPY* in determining the local instantaneous burn rate in a full-system simulation.

Rocstar includes two finite-element structural mechanics solvers, *Rocfrac* and *Rocsolid*. Both solvers feature an ALE formulation to account for the conversion of solid propellant into the gas phase. They handle large strains and rotations, can solve the 3-D heat conduction equation (new for *Rocfrac*), and include a variety (wider this year) of element types and constitutive models. *Rocsolid* has an implicit time integration scheme that uses the multigrid method and/or BiCGSTAB to solve the required linear systems efficiently in parallel. *Rocfrac* has an explicit time integration scheme, and an implicit one is under development. *Rocfrac* can include cohesive volumetric finite elements between ordinary elements to follow crack propagation.

Integration Framework and CS Services

The Integration Interface (center of Fig. 3.6.1) is a library (API) called *Roccom*. *Roccom* facilitates the exchange of data between different modules, including those written in different programming languages (C++, F90). By making a limited number of calls to *Roccom*, the physics applications gain access to a large number of useful CS service components in our integration framework (column of green boxes on the right-hand side of Fig. 3.6.1).

The orchestration module (red box in Fig. 3.6.1) called *Rocman* controls the execution of the physics applications, including initialization, coupled time stepping, output dumps, and stopping criteria. In *Rocstar* we adopted the "partitioned" approach to time stepping, in which each domain (solid, fluid) is evolved separately from the other domains for one system time step. After each module reaches the new system time level, it exchanges interface data with the other domains so that the system remains tightly coupled. *Rocman* was rewritten recently in C++ to provide much more flexibility and software reuse in implementing new coupling schemes. A jump condition derived from conservation of energy was added to enable simulations that include heat transfer between the fluid and solid domains. Additional time stepping schemes were investigated this year to provide improved stability and accuracy, and the Improved Staggered Scheme of Farhat was implemented.

The surface propagation module (*Rocprop*) computes the motion of the propellant surface as it regresses due to burning. *Rocprop* can be used in coupled simulations as well as fluids-only or solids-only calculations. It can be switched off for problems in which there is no significant loss of mass from the solid domain (fluid/structure interaction without burning, or evolution times << burn times). This year *Rocprop* includes a new, original, robust, and general surface propagation scheme called the "Face-Offsetting Method," which was developed at CSAR by Jiao.

The mesh modification schemes in *Rocstar* are in various stages of development. Mesh smoothing (without changing the number of mesh vertices) for unstructured meshes is accomplished in the *Rocmop* module through calls to the serial Mesquite package developed at Sandia National Laboratory. Each partition calls Mesquite concurrently, sending both real and ghost nodes. Mesquite smoothes only the interior nodes of these mesh partitions, so including the ghost nodes is essential to maintaining mesh quality. After Mesquite smoothes all partitions, the coordinates of vertices shared by multiple partitions are averaged to ensure that the meshes still match at partition boundaries. It is possible (but not usually necessary) to call Mesquite multiple times to alleviate any impact on mesh quality due to averaging shared nodes. Because the evolution equations in our solvers are formulated on moving grids, no solution transfer is required after mesh smoothing, although the amount that the mesh can change locally per call is evidently limited by a Courant-like stability criterion. Support for non-tetrahedral element types will soon be included in *Rocmop*, and even structured meshes will be smoothed by Mesquite. We have recently performed large production simulations (described below) using Mesquite in parallel.

Global remeshing and local mesh repair can now be performed using tools from Simmetrix, a company spun off from Professor Mark Shephard's group at Rensselaer Polytechnic Institute. The *Rocrem* module currently enables serial off-line remeshing, parallel solution transfer to the new mesh, and generation of all input files required to restart a simulation. The remeshing process is being automated, and a "warm" restart capability is being developed, so that *Rocstar* will be able to remesh without terminating the simulation or even writing any files to disk. We can also use Simmetrix tools to perform local mesh repair, operating only on selected mesh partitions. We are investigating ways to save wall clock time by taking advantage of the fact that much of a repaired mesh remains unchanged.

The solution transfer module called *Rocface* enables the physics applications to exchange interface quantities across non-matching meshes, which is essential to solving coupled fluid/structure interaction problems. The interpolation scheme is exactly conservative by construction, because it operates on an overlay mesh, which is a common refinement of the two meshes on either side of the interface. Each subdivision of the overlay mesh lies entirely within a cell face in both surface meshes. Moreover, interpolation errors are minimized in the least squares sense, leading to a scheme that has been demonstrated to be 20 times more accurate than recently published methods.

Rocstar automatically collects performance data for functions registered with *Roccom*, including physics application solution update times, data transfer times, output dump write times, etc. New this year: profiling at the subroutine, loop, or statement levels can be performed by inserting into the source code a small number of low-overhead calls to *Rocprof*.

Asynchronous Parallel I/O can be performed using *Rocpanda*. *Rocpanda* designates a user-specified number of processes as I/O servers, which collect data in the form of MPI messages from the compute processes, combine the data, and write it to disk in a manageable number of files in the desired format in the background as the simulation continues.

All modules in *Rocstar* use the MPI (Message Passing Interface) to pass messages between processes running in parallel. The modules are compatible with AMPI, an implementation of MPI developed at the University of Illinois that treats processes as user-level threads. There are two key benefits of AMPI for *Rocstar*: 1) the AMPI processes are "virtual" so that they can run on any number of actual CPUs, and 2) the virtual processes can be migrated from one CPU to another for dynamic load balancing. In performing large rocket simulations, we have used the first of these two features extensively to utilize available computational resources (fewer processors available than the number of partitions). Dynamic load balancing becomes important after the meshes have been refined or coarsened in some partition, due to either geometrical changes (e.g., propellant burning and deformation) or solution-based mesh adaptation.

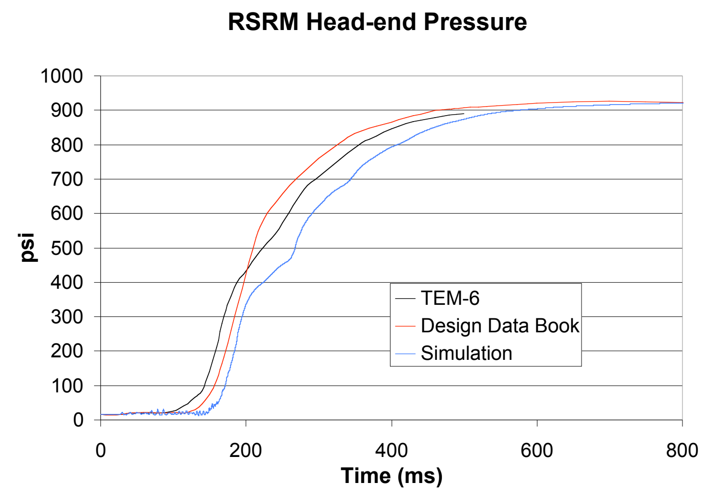

Fig. 3.6.2: RSRM pressure histories.

Simulations

NASA Reusable Solid Rocket Motor

We simulated ignition transients in the Space Shuttle Reusable Solid Rocket Motor (RSRM) using *Rocflu*, *Rocfrac*, and *RocburnPY*. This model geometry includes many details, such as precise inhibitor radii in joint slots and the "submerged" nozzle. The unstructured fluid mesh contains 4.5 M tetrahedral elements and is smoothed by calling Mesquite every few fluid time steps, which costs only 30% more run time than not moving the mesh at all. The gas is assumed to be inviscid, radiation is ignored, and a simple "film coefficient" heat transfer model is used for propellant heating. Huang's advanced constitutive model is used for the propellant.

Head-end pressure histories for our simulation, nominal values from the Space Shuttle Design Book, and the TEM-6 test firing are shown in Figure 3.6.2. Our 3-D simulation captures the important features in the ignition transient without resorting to numerous empirical physical models. These features include the delay between igniter firing and rapid pressurization, the slope (dP/dt) as it responds to the arrival of sound waves reflected from the nozzle, and the quasi-steady operating pressure. Agreement would be better if we had adopted a lower propellant ig-

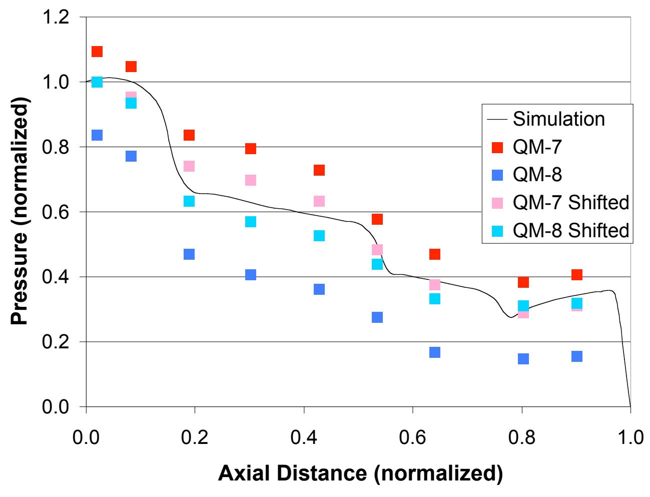

Fig. 3.6.3: RSRM axial pressure profile.

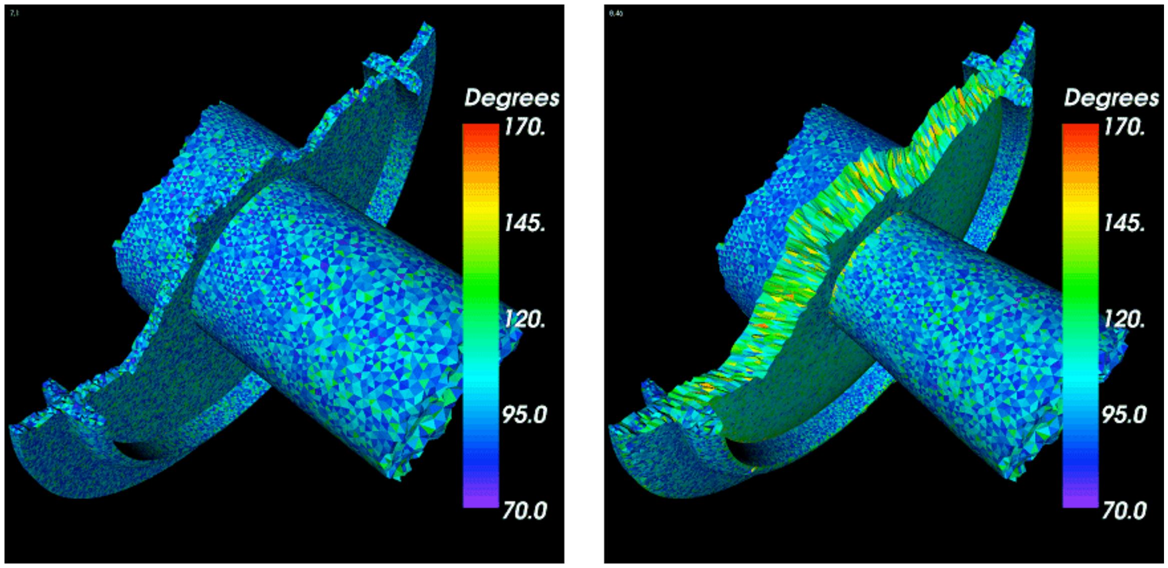

Fig. 3.6.4: Titan initial (left) and deformed (right) mesh quality at the joint slot.

nition temperature or a higher heat transfer coefficient. These quantities are known only to a significant degree of experimental uncertainty.

Prior to our 3-D simulations, ignition and flame spreading were poorly understood and mostly modeled empirically in 2-D (even in the star slots). It had been assumed that radiation was a major factor in igniting the propellant in the star slots. Our simulations demonstrate that convection occurs rapidly in the star slots, but not in the joint slots and submerged nozzle where cool gas is trapped.

Figure 3.6.3 shows the pressure as a function of position in the quasi-steady state, comparing our results to the Design Book and two test firings that were done at different ambient temperatures. The two actual test firings produced results that are remarkably different from each other. Even after we normalize the experimental pressure data and shift them so that all pressures agree at the head end, the experimental data differ widely in the rocket's mid-section. Note that the experimental data is derived from hoop strain measurements taken outside the rocket case, and conversion to internal pressures by the experimenters may be inaccurate. Our results agree well with the test firings even without including physics that could be important (turbulence, radiation, non-rigid case, etc.) because of the large experimental uncertainty. This means that in order to improve our simulations, the measurements taken during test firings need to be more detailed, and the properties of the particular rocket being fired need to be measured and used as input to the simulation. Nominal properties from the Design Book are not sufficient for precise prediction, since for example the specifications allow the burn rate to vary by +/- 5%, leading to quasi-steady head end pressure differences of +/- 20%.

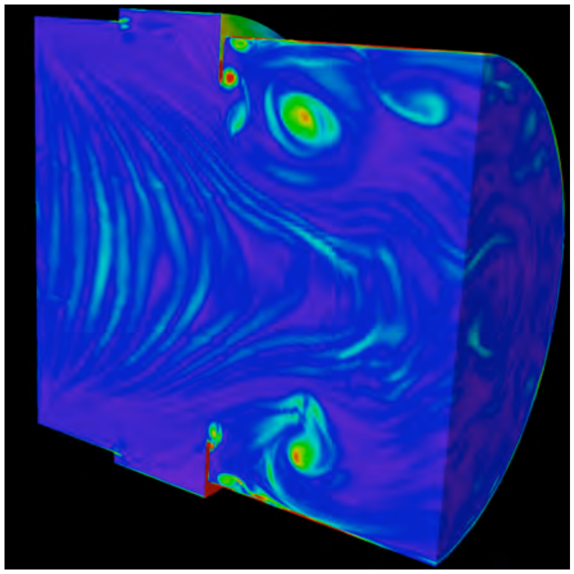

Fig. 3.6.5: Vortex shedding from the inhibitor at RSRM joint slot.

Titan IV

We used *Rocstar* to improve upon our previous 3-D simulations of propellant slumping in the Titan IV SRMU Prequalification Motor #1 (PQM-1). We are investigating an aeroelastic effect that occurs at joint slots where, for manufacturing reasons, there is a smaller propellant bore on the aft side. The thermal pressure is considerably higher inside the slot than it is just downstream of the slot, where the gas speed is much higher. This uneven load distribution tends to pull the propellant toward the axis of the rocket on the aft side of joint slots, possibly to the point of choking the flow of exhaust gasses. This effect caused the destruction of this motor on the test stand in 1991.

A frequently cited paper on the Titan failure claims to have reproduced the propellant slumping and failure to a high degree of accuracy using linear elastic material models in 2-D. When we make the same assumptions in our 3-D simulations, we obtain very different pressure histories because the propellant deforms much sooner than it did in the real rocket. We conclude that the propellant properties deduced using simple material models are incorrect, and that the older paper's results are of questionable validity.

A new non-linear viscoelastic propellant material model, developed by Sofronis, includes the effect of voids and dewetting, and was recently implemented in our implicit structural dynamics solver *Rocsolid*. Additional simulations used our explicit solver *Rocfrac* with Huang's advanced propellant model.

Maintaining mesh quality in the fluid domain remains a major challenge in simulating the Titan failure. We performed some Titan simulations using *Rocflu* and Mesquite. In other Titan simulations, we used *Rocflo* with an improved solver-specific mesh motion scheme in which the block corners are smoothed first, and then nodes along edges, and then surface and interior nodes, so that the deformation is propagated as rapidly as possible between neighboring blocks. This structured mesh motion scheme is undergoing further development, especially since we have no hope of automatically remeshing a structured grid in the way that we can a tetrahedral grid.

Figure 3.6.4 shows maximum dihedral angles in fluid elements only for a region near the center joint slot from a *Rocflu/Rocfrac* simulation that modeled the full rocket in 3-D. Some mesh blocks are not included, allowing a view of the mesh in the slot. This portion of the rocket is nearly axisymmetric. There are initially four tetrahedral elements across the joint slot, and as the slot opens due to the load on the propellant, these slot elements are stretched several times their initial length, primarily in the axial direction. Mesquite does a good job maintaining high mesh quality until shortly after the output time of the right hand panel in Figure 3.6.4. Simulating even more slumping requires remeshing, a capability described above that is nearly ready for *Rocstar* production runs.

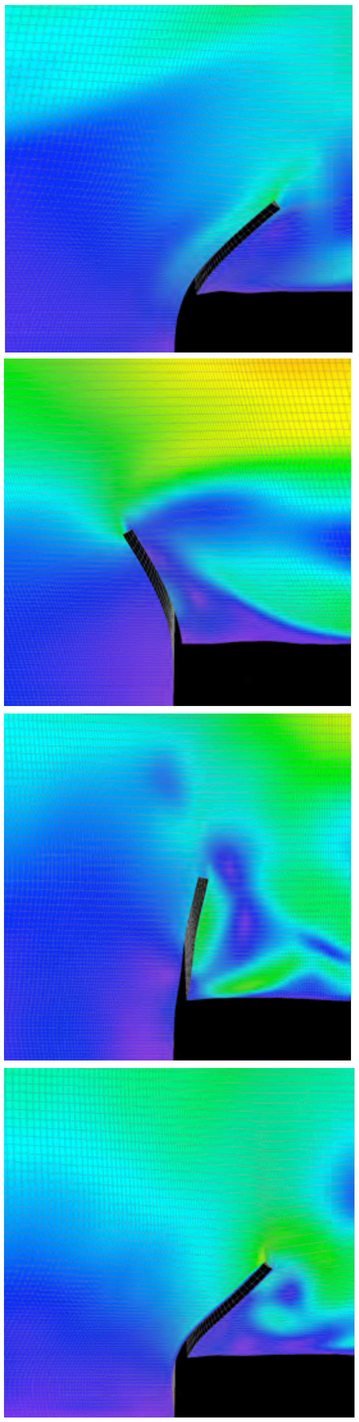
Fig. 3.6.6: Velocity magnitudes near flexible inhibitor (7. 0, 17.5, 23.0, 26.0 ms).

RSRM Flexible Inhibitor

We performed fully coupled *Rocstar* 3 simulations of gas flow in the vicinity of a flexible inhibitor protruding into the combustion chamber of the RSRM, including LES turbulence (dynamic Smagorinsky model). The geometry we are now using matches the RSRM at 100 seconds after ignition, when the inhibitor protrudes 25 to 30 cm into the combustion chamber. We model a section of the rocket near the center joint slot in these *Rocflo/Rocsolid* simulations, using 2 million structured fluid elements. The inflow conditions make use of velocity and pressure profiles computed in our full RSRM simulations described above. The high resolution of these simulations allows accurate computations of turbulent flow.

Figure 3.6.5 depicts the vorticity magnitude after a turbulent flow has been established in a simulation with a rigid inhibitor. A flexible inhibitor would swing back and forth due to gas pressure forces, causing the vortex shedding frequency to fall and rise, introducing perturbations into the flow that could induce pressure oscillations and even drive instabilities in the overall rocket fluid flow.

Figure 3.6.6 shows the deflection of the inhibitor at four different times from a run without turbulence (inviscid), which was made while developing *Rocflo's* latest mesh motion scheme. The inhibitor material is 100 times stiffer than the value provided by the rocket manufacturer, and yet the inhibitor oscillates violently, presenting a daunting challenge to any mesh motion scheme. If the actual inhibitor is as compliant as indicated by the provided properties, it surely comes in contact with the burning propellant surface downstream. The papers in the literature had predicted that the inhibitors would be deflected by the gas flow, but only small oscillations about an equilibrium angle of deflection were anticipated.

Superseismic Shock

The superseismic shock is a coupled fluid-structure interaction problem with a shock wave propagating along the fluid-solid interface faster than any wave speed in the solid. The interaction of the shock in the fluid with the solid boundary yields an asymptotic solution for angles δ and β in Figure 3.6.7. We compare *Rocstar* results to this solution to verify the coupled code.

Figure 3.6.7 shows a *Rocstar* snapshot of a Mach 3.0 shock when it has traversed 90% of the fluid domain. Table 3.6.1 compares *Rocstar* simulation results with analytic results of the β angle for three different shock intensities.

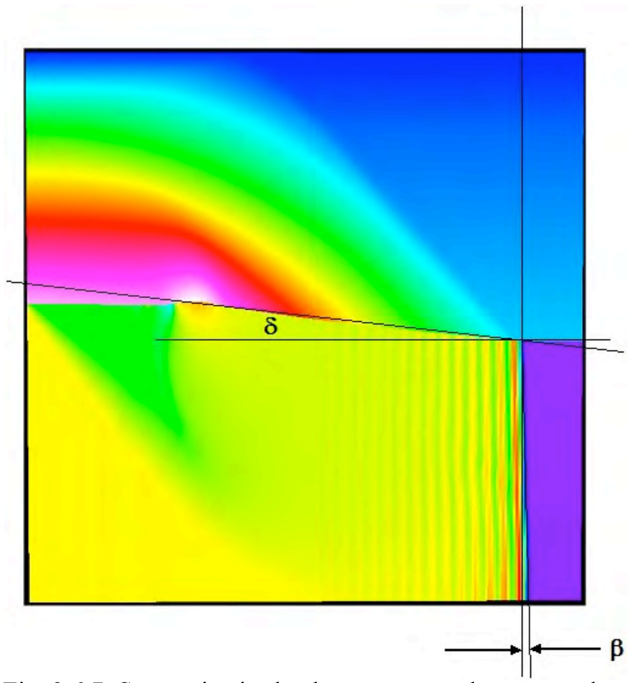

Fig. 3.6.7: Superseismic shock geometry and pressure plot for Mach 3.0

Table 3.6.1: Superseismic Shock Results for Three Different Mach Numbers with *Rocstar* 3

Mach	β Theory	β Simulation
2.5	-1.7982	-1.73±0.04
3.0	-2.1214	-2.12±0.03
3.5	-2.3041	-2.28±0.12

Elastic Piston

Another quasi-1D fluid-solid interaction problem with an analytical asymptotic solution being used for verification involves the transient coupling of an inviscid incompressible fluid and an elastic solid block, which acts as a piston. The interface boundary is assumed to be impermeable, nonreactive, and adiabatic. The system is initially at rest, with no load on the solid and quiescent conditions everywhere in the fluid domain. At time t=0, the pressure load is applied to the solid material, resulting in the motion of the interface boundary with velocity v_I, which is constant in the analytical solution. A dilatational wave propagates through the solid domain, and expansion fans propagate through the fluid domain. The analytical solution is valid until waves reach any domain boundary.

Two coupled *Rocstar* simulations were performed using *Rocfrac* to compute the solid response, and exercising both the unstructured- and structured-mesh fluid solvers. Figure 3.6.8 shows wave motion progressing in the solid on the left, and midway through the fluid domain on the right. The solid is linear elastic material, and the fluid is a perfect gas. For the initial conditions used, the wave speed in the solid is 62.4 m/s, and the speed of sound in the fluid is 338 m/s. The implicit equation for the interface boundary velocity yields an analytical value of 0.57 m/s, valid for approximately 3 milliseconds of simulation time. Figure 3.6.9 shows a comparison of preliminary *Rocstar* results and the analytical solution. Agreement improves after a transient due to the initial conditions subsides. The structured fluid solver used a low-order approximation at the interface, causing larger discrepancies than the unstructured solver.

Mesh Motion – Motor 13

An addition to the capabilities of *Rocstar* during the past year is *Rocprop*'s ability to regress burning surfaces in a fluid-only simulation just as they would in a fully-coupled simulation (without deformaition). In many situations, the propellant deformation is small and has little effect on the chamber pressure. One such rocket is a laboratory-scale rocket identified as Motor 13 for which basic geometry and pressure data is available. Figure 3.6.10 shows a side-by side comparison of the propellant at 0.0 and 1.8 seconds after ignition (full burn is 6.0 seconds). In this validation problem, regression of the propellant surface leads to a larger burning surface area and a corresponding increase in pressure. One such rocket is a laboratory-scale rocket identified as Motor 13 for which basic geometry and pressure data is available. Figure 3.6.10 shows a side-by side comparison of the propellant at 0.0 and 1.8 seconds after ignition (full burn is 6.0 seconds). In this validation problem, regression of the propellant surface leads to a larger burning surface area and a corresponding increase in pressure. Figure 3.6.11 shows the experimental pressure and the simulation results. The *Rocstar* results are somewhat higher than the experimental results due to our adopted flame temperature value. Nevertheless, the two pressure curves are essentially parallel, and therefore surface regression is progressing very much as it did in the test firing.

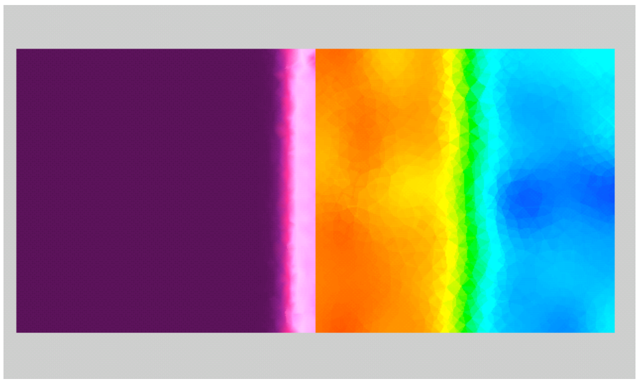

Fig. 3.6.8. Wave motion in elastic piston solid and fluid domains

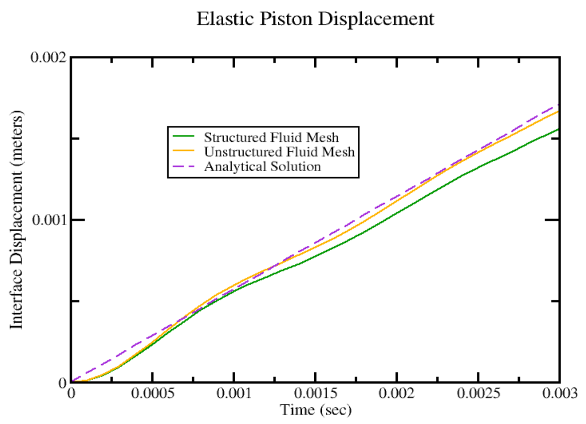

Fig. 3.6.9: *Rocflo* and *Rocflu* elastic piston simulation interface displacement comparison with theory

Flexible Panel in Shock Tube

The literature describes an experimental shock-tube study in which a Mach 1.21 shock in normal air is introduced at one end. The shock travels down the tube until it encounters a thin, flat, steel panel projecting vertically into the flow. The panel is fixed at the bottom end, and free at the top and sides. Experimental measurements and observations of the shock profiles and plate movement are provided, as well as results from the author's own fluid-structure interaction simulations. This problem thus forms a good validation problem for the coupled fluid/structure interaction modes in *Rocstar*. Figure 3.6.12 shows the geometry for this problem. Figure 3.6.13 shows the *Rocstar* pressure at 0.3 ms after the wave impacts the flexible panel. The qualitative behavior of the transmitted and reflected shocks in the simulation is visually identical to the shadowgraphs given in the paper.

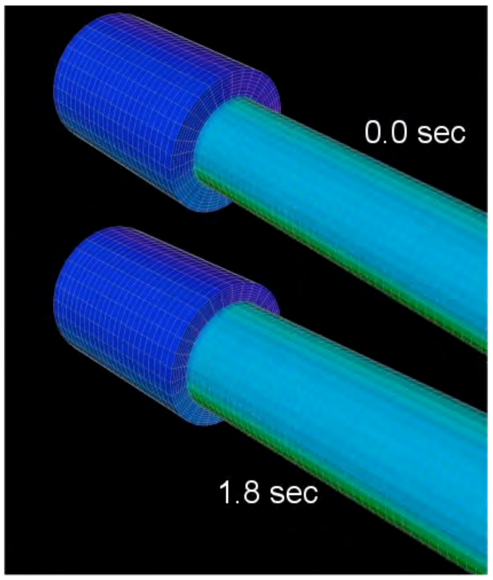

Fig. 3.6.10: Surface regression in Motor 13

As can be seen in Figure 3.6.12, there is also a pressure transducer near the chamber ceiling just upstream of the plate. A plot of the pressure at this position computed by *Rocstar* versus the experimental results is shown as Figure 3.6.14. From this figure, it is obvious when the shock arrives at the plate. Following the paper, this is taken as time zero for the analysis.

The pressure profile of the simulation is very similar to the experiment in timing and features. It is, however, slightly high in average pressure, as were the simulation results reported in the paper. This discrepancy is caused by a difference in the boundary conditions used in the simulation compared to the experiment. More details were recently obtained on the experimental initial conditions in preparation for further simulations.

The results shown above characterize the fluid conditions due to the shock traveling down the tube and impacting the panel. The second result of interest is the response of the 50 mm long, 1mm thick steel panel to the applied loads. The results from the paper show an experimental maximum tip displacement of 6.2±0.4 mm for the first "swing" of the panel after the shock impacts it. The experimental period of oscillation is approximately 4.8 ms.

Figure 3.6.15 shows the displacement of the tip of the panel. The maximum displacement calculated during the *Rocstar* simulation is 6.68 mm, which is very close to the experimental maximum. As the figure shows, the panel's movement through its first half-period of motion is captured quite well by *Rocstar*. Advancing this simulation further is planned.

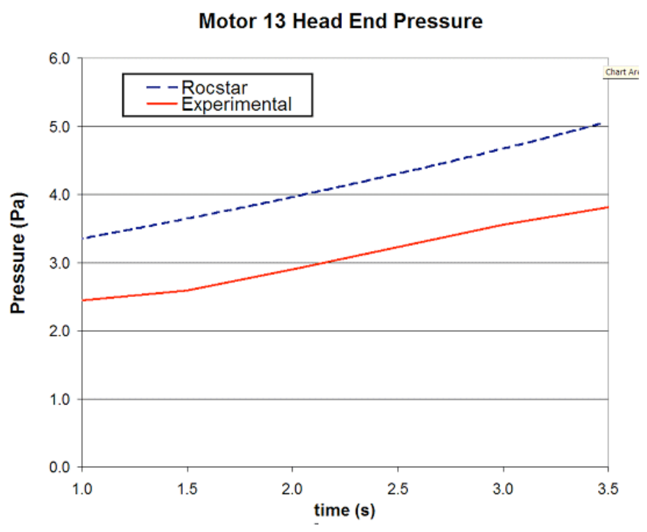

Fig. 3.6.11: Motor 13 burnback pressure comparison

Software Engineering Initiatives

Daily automated building of both the integrated *Rocstar* code and the fluid modules on multiple platforms are performed by the *Rocbuild* system. The integrated code is built with and without the the Charm++ package, and various physics options are enabled or disabled. Both the *Rocflo* and *Rocflu* codes are built in a total of 36 different combinations to ensure timely identification of cross-platform compilation errors. Machines similar to ASC systems are included, as well as the new UIUC *Turing* cluster.

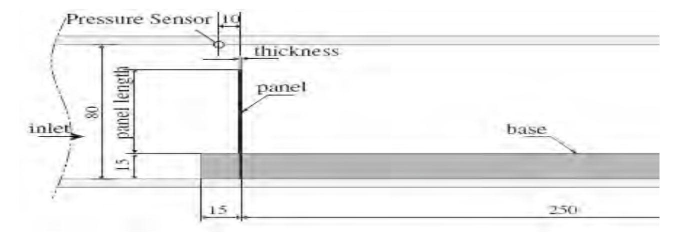

Fig. 3.6.12: Flexible panel in shock tube geometry

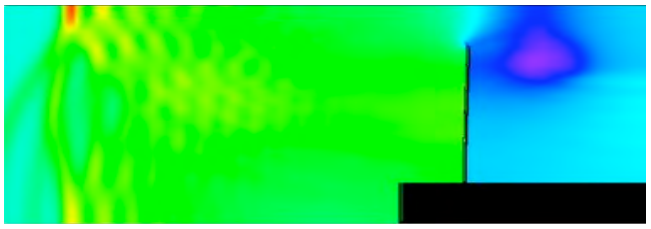

Fig. 3.6.13: Pressure at 0.30 ms after shock impact on plate showing transmitted and reflected shocks.

The *Roctest* automated regression testing module has been expanded to run six builds on six different regression test problems nightly on the UIUC Apple cluster to provide confidence that the most recent code has not been seriously compromised by check-ins within the past 24 hours. A new utility called *Rocdiff* is invoked by *Roctest* to compare each of the regression test runs against a set of archived "gold" results, and to report any differences encountered. *Rocdiff* performs node-by-node comparison of all physics results in the gold result files.

Over the past year, the *Rocprep* dataset preparation tool has matured, and is now being used by a number of CSAR staff and graduate students who run *Rocstar*. *Rocprep* automates preprocessing of coupled, *Rocstar* format datasets. *Rocprep* reads the basic dataset information from a new archival format called a Native Data Archive (NDA). NDAs are designed to include, for any specific model, multiple sets of input data and multiple grids, so that an analyst may pick any available grid and couple it with any compatible set of physics data. Rocprep has decreased dataset preparation time significantly, although it cannot assist with initial physics data collection or grid preparation. Documentation of our broad set of NDAs are being posted on an internal web site for users.

Integration Future Plans

We will continue to enhance the capabilities of *Rocstar* in many ways to enable it to address more scientifically interesting problems with changing geometries. Since the solvers use body-fitted grids, maintaining good mesh quality will remain a key issue. We will enable mesh smoothing by Mesquite for elements other than tetrahedrons, including structured meshes in *Rocflo*, although *Rocflo's* internal mesh motion scheme will also continue to be improved. Remeshing and local mesh repair using Simmetrix tools will be completed, integrated, parallelized, and automated to a significant extent.

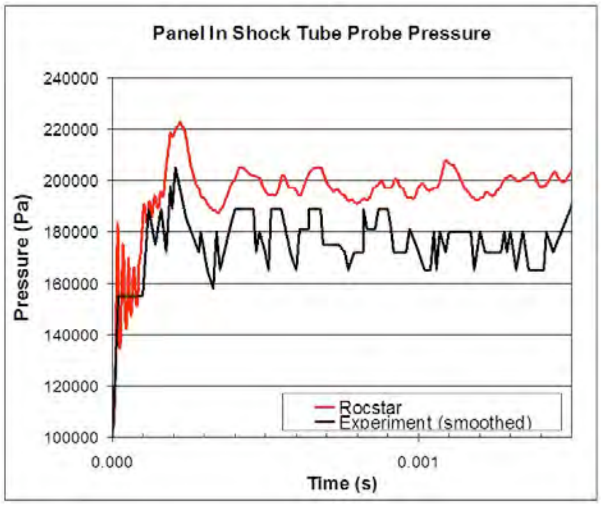

Fig. 3.6.14: Probe pressure versus time

The Titan and inhibitor simulations will continue to serve as test beds for mesh improvement schemes. More accurate material properties, including stress-strain curves, particularly at high strain rates, will be obtained for the materials in these rockets, so that we will be able to match the Titan pressure history closer to the point of failure.

Our newly-proven time zooming scheme will enable us to burn most of the propellant in much less time (by a factor of 20 to 200), making a detailed full burn of the RSRM feasible for the first time next year. We will carefully formulate a time zooming algorithm in the solid domain

Additional coupling schemes will be developed as required to apply *Rocstar* to wider areas of scientific and engineering interest. More efficient and stable time stepping schemes will be developed and integrated into *Rocman.*

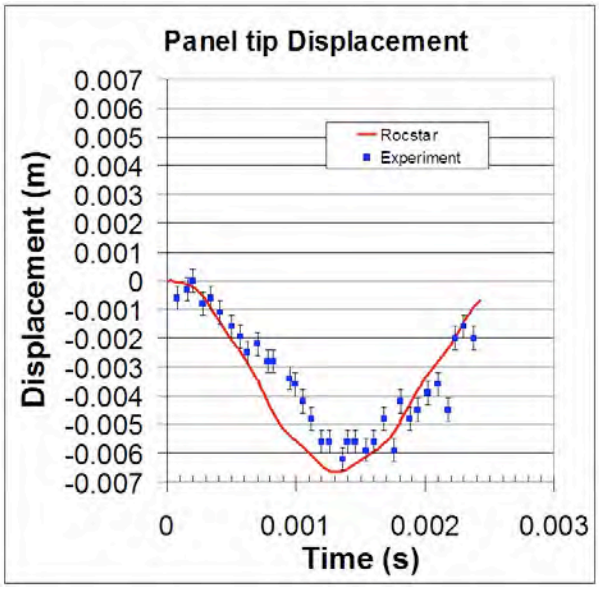

Fig. 3.6.15: Panel tip displacement with time

Code verification will be performed using manufactured solutions to determine the actual order of convergence. We will devise manufactured solutions that exercise more of the huge number of code features available in *Rocstar*.

The *Roctest* program will be expanded to perform regression tests on multiple platforms, and to increase the number of different problems run. Additional test problems will be identified for validation of *Rocstar*. All new and existing input data sets will be archived in the NDA format and documented. The V&V process will be expanded to include consideration of Error Estimation and Uncertainty Quantification techniques as they are identified and shown to be applicable to CSAR codes and simulations.

Development plans also include a consolidated input GUI for generating the many text input files for the physics modules in *Rocstar*. This tool will greatly enhance the ease-of-use of the code suite. The GUI will include features such as intelligent groupings of input parameters, with frequently used values being prominent and infrequently used values hidden in lower menus or tabs. Also, default values would be supplied or automatically calculated for most inputs, freeing the user to focus on only those parameters of interest.