AIAA-2001-3954

COUPLED FLUID-STRUCTURE 3-D SOLID ROCKET MOTOR SIMULATIONS

R. Fiedler[*], X. Jiao[†], A. Namazifard[‡], A. Haselbacher[‡], F. Najjar[‡], and I. D. Parsons[§]
Center for Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign, Urbana, IL 61801
http://www.csar.uiuc.edu

Abstract

We describe our numerical method for three-dimensional simulations of solid rocket motors in which the internal gas dynamics, the combustion of the propellant, and the structural response are fully coupled. The combustion zone is treated as a thin layer using appropriate jump conditions, and the regression rate is determined using a nonlinear dynamic combustion model. An Arbitrary Lagrangian-Eulerian formulation is used in the gas dynamics and structural mechanics modules to follow the regression of the propellant. We demonstrate the parallel scalability of our ALE implementation and its ability to handle significant burn back of the propellant on a model problem with a very high burn rate.

Introduction

Detailed 3-D simulations are routinely performed by aerospace engineers to analyze virtual prototypes of aircraft, but the US rocket industry typically employs one- or two-dimensional models during the design phase, usually treating each component as an isolated system. While a great deal of physics can be included in such calculations, the models adopted are often based on measurements of simpler, but not necessarily closely related systems, rather than on first principles.

The primary goal of the Center for Simulation of Advanced Rockets is to perform detailed, whole-system simulations, making use of science-based models rather than empirical relations whenever possible[1,2]. Obtaining accurate numerical solutions to such a complex problem requires enormous computational resources. Moreover, the physics modules (gas dynamics, structural mechanics, etc.) must all run efficiently in parallel on many processors and operate in concert to solve a tightly coupled system.

This paper describes version 2 of the GEN1 rocket simulation package developed at CSAR, with an emphasis on improvements from version 1.0[3].

Overview of the Coupled Simulation Package

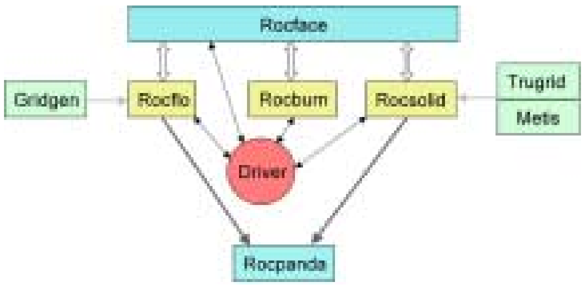

Figure 1. GEN1 Schematic

Figure 1 shows the overall structure of GEN1 version 2. The system code consists of the physics modules (Rocflo, Rocburn, Rocface), infrastructure modules to pass data between the physics modules and to dump output (Rocface, Rocpanda), and a main program to control the execution of all components (Driver). The system code runs as a single executable on each processor of a parallel computer. Problem setup is performed off-line using separate applications (Gridgen, Truegrid, Metis).

In developing our integrated simulation software, we adopted a partitioned approach, in which each physics module solves a problem within a specified region of space. The modules are executed one at a time, and updated surface values from one module are passed to the others for use as boundary conditions. Complete coupling of all modules is enforced by a

[*] Technical Program Manager
[†] Visiting Scholar
[‡] Research Scientist
[§] Associate Professor of Civil and Environmental Engineering, University of Illinois at Urbana

predictor-corrector solution procedure, in which the overall system time step is repeated until changes in the solution from one iteration to the next are within prescribed tolerances. This partitioned approach has allowed us to couple together pre-existing physics applications with a limited amount of modification and to continue to develop them independently.

In the next several subsections, we describe the individual components of the GEN1 version 2 package.

Gas Dynamics Solver: Rocflo

Rocflo is a Computational Fluid Dynamics application designed to simulate solid-rocket motor core flows. A structured multi-block finite-volume approach with cell-centered data storage is adopted. The compressible Navier-Stokes equations are solved in Cartesian coordinates using an Arbitrary Lagrangian-Eulerian (ALE) formulation to allow for boundary movement due to deformation and burning of the propellant. For deforming meshes, the cell geometric properties are computed such that the Geometric Conservation Law (GCL)[4] is satisfied.

The spatial discretization schemes implemented in Rocflo are the central scheme with artificial dissipation[5] and two second-order TVD schemes: Roe's flux-difference splitting scheme[6] and Yee's symmetric TVD scheme[7]. Several limiter functions are available in Rocflo. The computations shown in this paper employed Roe's upwind scheme with second-order accuracy and the "minmod" limiter function. The viscous fluxes are computed using gradients calculated at face centers.

The discrete equations are integrated in time using an explicit multistage Runge-Kutta method. For unsteady computations, a two-stage method is commonly chosen, where the time step is given by the smallest value of the largest stable time step among all fluid cells.

Rocflo is implemented in Fortran 90. User-defined data types and pointers are employed extensively to ensure a clean code structure. As a result, each block may be regarded as an object and contains all data relevant to that block, including nodal coordinates, face vectors, cell volumes, and the state vectors. This form of data encapsulation also facilitates the mapping and migration of blocks among processors. To ensure high FLOP rates, computationally intensive parts of the code are written in Fortran 77-style.

The parallel implementation of Rocflo is based on the block topology, where each processor may own one or more blocks. After each solution update, blocks residing on the same processor simply copy boundary data while those on different blocks use "persistent communication" calls in the MPI message passing library.

The initial fluids mesh is created using the Gridgen mesh generation package[8] from Pointwise, Inc. The mesh is subsequently partitioned for parallel computation using a tool developed at CSAR by Orion Sky Lawlor and Mark Brandyberry. This tool also produces an input file for Rocflo that describes the block connectivity and boundary conditions.

More detailed information on Rocflo, including results for validation cases and parallel performance studies for fixed and scaled problems, is provided elsewhere[9]. Rocflo has recently been extended to include aluminum droplets, smoke particles, and various chemical species[10]. However, the results presented here involve strictly single-phase flows.

Structural Mechanics Solver: Rocsolid

In order to compute the structural response of the propellant, case, liner, and nozzle, we developed Rocsolid, a structural analysis code that employs a finite element discretization of the solid components using unstructured hexahedral meshes. Dynamic problems are solved using the implicit Newmark time integrator[11].

Rocsolid offers two types of linear system solvers for the matrix equations encountered within the Newton iterations at each time step. For problems in which no interface moves through the material (e.g., burn times so short that the regression of the propellant is negligible), the linear system resulting from the equations of motion is symmetric and can be solved most efficiently using a scalable parallel multigrid method[12]. For problems with significant propellant burn-back, the equations of motion are expressed in a parametric domain which is mapped to an evolving undeformed reference configuration[13]. The resulting linear system is non-symmetric, and a BiCGSTAB method is used. Although this solver is parallel, it is inherently less scalable than multigrid because, unlike multigrid, the number of BiCGSTAB iterations required to obtain a solution to within a specified tolerance increases with the size of the problem (i.e., the total number of finite elements).

Examination of the multigrid and BiCGSTAB algorithms indicates that all of the operations can be performed independently on domains partitioned for parallel execution. In particular, the main components of the algorithm are matrix-vector multiplications that can be efficiently implemented element-by-element. Communication between partitions (processors) is required only during matrix-vector multiplications, scalar products and fine-to-coarse mesh restriction. Matrix-free element computations reduce the storage and the time requirements of our implementation. Rocsolid is written in Fortran 90, and uses MPI to pass messages between processors.

Multigrid methods require a hierarchy of increasingly finer meshes. We use TrueGrid[14] from XYZ Scientific Applications, Inc. to produce a sequence of nested, uniformly refined hexahedral meshes, which allows us to model complex parts. Mesh partitioning is performed on the coarsest mesh using Metis[15]. to achieve load balance between the processors. Uniform refinement of the coarsest mesh partitions produces the required partitions on the finer meshes. Thus, element load balance is maintained throughout the mesh hierarchy, although the resulting communication pattern may not be optimum.

Burn Rate: Rocburn

Rocburn (implemented in GEN1 by K. C. Tang) computes the unsteady burning rate of solid propellant using a nonlinear dynamic solid propellant combustion model. The theory of Zeldovich and Novozhilov (ZN) is used in combination with the Flame Modeling (FM) approach[16,17,18] in the model. For homogeneous solid propellant combustion, WSB flame modeling[19] is used. To simulate the combustion of AP composite propellant, an empirical expression replaces premixed WSB flame modeling[20].

At each fluid cell face on the burning propellant surface, the one-dimensional (into the propellant) unsteady energy equation in the condensed phase and the unsteady burning rate eigenvalue are solved for a prescribed unsteady pressure. Under rapid pressurization rates, the unsteady dynamic burning rate can be significantly different from the burning rate predicted by the quasi-steady combustion model due to the thermal relaxation effect in the solid[20,21].

One important undesirable motor behavior is the initial "ignition" pressurization spike. The "ignition" spike commonly observed in motors with small L* (ratio of free chamber volume to nozzle throat area) was usually attributed to erosive burning or igniter mass flux[22,23]. Using the nonlinear dynamic combustion model, the "ignition" pressure spike has been simulated in 0-D for homogeneous propellant combustion by Tang and Brewster[21] without including erosive burning and igniter mass flux effects. Similarly, for AP composite propellant, the "ignition" pressure spike in the NAWC tactical motor #13 test data[24] is also well captured by a 0-D calculation by Tang and Brewster[20]. These results suggest that the dynamic burning effect may play a significant role in producing the "ignition" pressure spike. Even in the absence of a pressurization spike, these results suggest that nonlinear, pressurization-rate dependent dynamic burning may contribute to rapid pressurization in some motors[21].

Data Transfer Across Interfaces: Rocface

The interface code is responsible for transferring data at the fluid-solid interface, where the two meshes are potentially non-matching. To perform the data transfer, we must first identify the geometric relationship between these non-matching interface meshes. In our current implementation, we use a mesh association algorithm to locate the closest solid element for each fluid nodal point on the interface[25], and use this result to transfer motion and loads at the interface. The mesh association algorithm traverses the fluid nodes and solid elements from neighbor to neighbor so that the closest elements are located quickly. After mesh association, we transfer displacements and velocities from solids to fluids by interpolating the values for each fluid nodal point at its closest point on the solid interface, and scatter the fluid nodal forces to solid nodes. The same set of coefficients is used for both load and motion transfer, and as a consequence, guarantees global conservation of energy[26].

With the ALE formulation, it is also necessary to transfer the regression rate from fluids to solids, and more general interpolation algorithms are required. We have developed new data transfer algorithms and are incorporating them into our simulation code. Before this new code is available, for burning fluid-solid interfaces, we limit ourselves to matching interface meshes. For this case, the mesh association algorithm is used to identify the correspondence of the nodes between the fluid and solid interface meshes, and regression rate is essentially copied from fluid nodes to their corresponding solid nodes. The restriction of matching meshes at burning interfaces will be eliminated with the new interface code.

In the new version, we construct a *common subdivision* of two non-matching meshes and a collection of more accurate and more general data transfer algorithms is built on top of the common subdivision. A common subdivision of two meshes is a finer mesh such that each face or edge of the two given meshes is partitioned into a set of faces and edges of the finer mesh. We have developed an efficient algorithm for computing common subdivision[27]. The common subdivision defines, and allows efficient query of, a unique nearby corresponding point on one surface for every point on the other, and enables conservative data transfer between meshes. In particular, we have developed a set of least-squares data transfer algorithms which are not only conservative, but also minimize the overall error[28].

Since the fluid and solid meshes are distributed across multiple processors, the interface code must handle distributed meshes. In our current parallel implementation, we first compute the bounding boxes of the connected components of the partitions of the solid mesh and of the blocks of the fluid mesh. These bounding boxes are compared to provide a quick estimate of the adjacency between the solid partitions and the fluid blocks. Then the solid partitions adjacent to a fluid block are shipped to the processor that owns the fluid block, and are connected together to form a new mesh. The sequential mesh association algorithm is then applied on all processors in parallel. As a by-product of mesh association, a more efficient communication pattern is obtained, which is used to exchange field variables between processors. The parallelization of the new Rocface takes an approach similar to the one described here.

Orchestration: Driver

The driver or main program orchestrates the execution of the other modules in the GEN1 code. At startup, it calls intialization routines in the physics modules, and then prepares Rocface to exchange data across interfaces.

The bulk of the main program is a loop over system time steps. This loop implements our predictor-corrector temporal coupling scheme[29]. On the first predictor-corrector iteration, Rocflo estimates the velocity of the solid surface from the previous system time level and takes a prescribed number of explicit fluids time steps. It then passes surface forces (and the new position of any burning surfaces) at the advanced time level to Rocsolid through Rocface. Using the surface forces as a boundary condition, Rocsolid takes one implicit time step to catch up with Rocflo, and then passes the new surface position and velocity to Rocflo through Rocface. On subsequent predictor-corrector iterations, Rocflo uses the surface velocity received from Rocsolid in the previous iteration and repeats its explicit time steps. Next, Rocflo passes the values (at the advanced time) of surface forces (and burning surface positions) to Rocsolid, which then repeats its implicit time step. At this point Rocface performs a convergence test on the surface forces, positions, velocities, and regression rates. If the relative difference of any of these quantities compared to the previous iteration exceeds the specified tolerance, another iteration is performed.

For problems without regressing surfaces, we typically take 10 fluids time steps to one system time step, and the predictor-corrector cycle converges after 2 or 3 iterations. For problems with regressing surfaces, 3 predictor-corrector iterations are frequently required. Because the BiCGSTAB solver is significantly more CPU intensive than the multigrid solver, we set the ratio of system to fluids time steps to as large a value as possible, under the constraints that the predictor-corrector cycle should converge in 3 iterations or less and the BiCGSTAB solver should converge in 50 or fewer iterations. We find that the best value of this ratio is problem dependent, and often changes as the calculation proceeds. We are experimenting with adaptively adjusting this ratio within GEN1.

Parallel Output: Rocpanda

The main program calls output routines in the physics modules when the system time step has advanced beyond the "*n*th" evenly spaced time interval, where "*n*" ranges from zero to typically hundreds of output dump times.

In a parallel computation, the simplest way to dump output is to have each processor write its data to a separate file, and use a visualization tool that can merge the blocks into a single image. If the problem and the number of processors is very large, a huge amount of data is written to disk at once, overwhelming the I/O system and delaying the calculation for many minutes. Moreover, transferring many Gigabytes worth of files to another system for storage or post processing can be very time consuming. To circumvent these problems, we use Rocpanda[30].

Rocpanda is an extension of the PANDA[31] library for CSAR's rocket simulation code. It uses additional processors (I/O nodes) to collect output data from the compute processors and write it to disk while the compute processors proceed with the calculation. This helps because it usually takes significantly less time to send the data over the internal network of a large parallel machine than it does to write it to hard disk. The amount of memory on the I/O nodes should be enough to hold one snapshot if there is to be no waiting for writes to disk. The I/O nodes can also combine the data into a more manageable number of output files. Another useful feature of the PANDA libray that will soon be added to Rocpanda is background file transfers, which allows the snapshots to be moved to mass storage and/or another system for post-processing as soon as the snapshots are written, rather than after the calculation has completed. Since the computation may take many days of wall clock time, all but the final output dump could be transferred automatically by the time the calculation is completed.

Visualization: Rocketeer

CSAR has developed a powerful, general purpose tool for scientific visualization called Rocketeer[32]. Rocketeer is written in C++ and based on the Visualization Toolkit[33,34], which uses OpenGL to exploit hardware graphics acceleration. The user interface employs wxWindows[35] for portability across platforms. Rocketeer executables for linux, Sun Solaris 2.7, and Microsoft Windows can be downloaded from the Rocketeer Web page.

Rocketeer is designed to read field data in HDF format[36] defined on structured or unstructured single block or multiblock grids. It also can display data defined on sets of points in space that do not constitute a grid. Rocketeer automatically merges grid blocks to produce seamless images from multiblock data files. Unstructured grids handled by Rocketeer may consist of tetrahedra, prisms, pyramids, hexahedra, etc. Surface meshes composed of triangles, quadrilaterals, etc., are also supported.

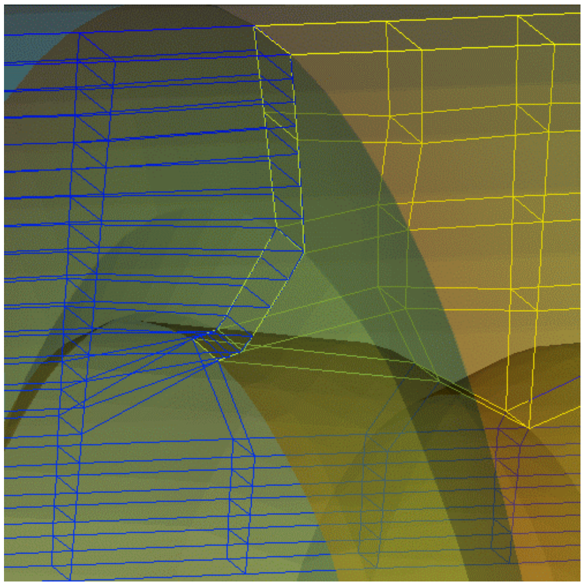

Figure 2. Rocketeer Visualization of Bad Mesh

Rocketeer can display field data using a variety of techniques, including surface plots in which values of a scalar variable determine the color, colored isosurfaces, and/or slices along the x, y, and/or z axes. Both 2-D surface and 3-D interior grids can be shown, with several choices available for selecting a small portion of a 3-D mesh (see Figure 2). Clipping planes can be used to cut an image along the x, y, and/or z axis, and the opacity of all objects can be varied to allow the user to see more deeply into a 3-D data set. Multiple windows can be open on the screen at the same time, and the camera position can be saved and loaded to assist the user in comparing similar data sets.

Scalar and vector quantities can be depicted using glyphs (usually spheres for scalars and oriented cones for vectors). The sizes of the glyphs can be uniform or they can vary with the value of some variable, such as the radius of aluminum particles or the magnitude of the velocity vectors (see Figure 3, in which the color of the cones indicates the speed). For more details and examples, see the extensive on-line User's Guide.

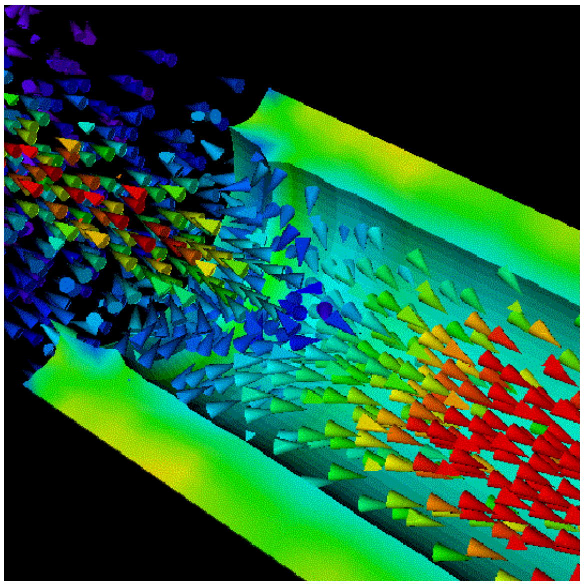

Figure 3. Rocketeer Velocity Glyphs

An MPI parallel batch mode version of Rocketeer called Voyager has recently been developed[37]. Each CPU processes in parallel a subset of a series of snapshots from a simulation. The camera position and list of graphics operations to be performed are saved during an interactive session of Rocketeer and read in by Voyager. Voyager has nearly all of the features of the interactive version, but saves images to files rather than displaying them. We have obtained parallel speedups over 72/96 when each of 96 nodes in our linux cluster processes one snapshot on its local disk. When the snapshots reside on a shared file system, contention for I/O bandwidth hurts scalability, but if the run is repeated (as might be done when the user adjusts the camera position, isosurface levels, etc.), much of the data saved automatically by the system in disk caches is reused and the runtime is nearly is fast as it is when the data resides on local disk (see Figure 4). After Voyager creates the images, they can be converted, for example, into a GIF animation file using ImageMagick[38].

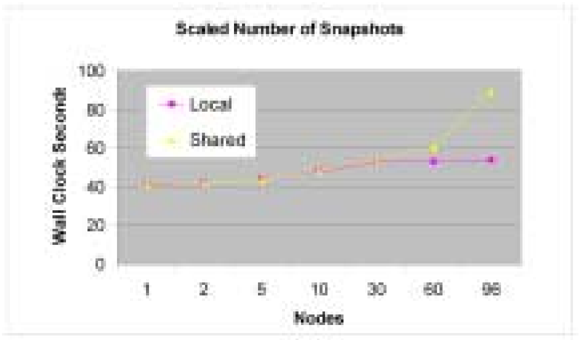

Figure 4. Voyager Parallel Performance

Simulations

In this section we present several results obtained using the latest capabilities in the GEN1 v2.0 package, as well as describe some ongoing simulations.

Scalability of the ALE Algorithm

To measure the parallel scalability of GEN1 v2.0 with regressing boundaries and dynamic burning, we solve a simplified 3-D rocket problem in which the geometry has axial symmetry (see Figure 5). At the head end, the solid has no applied load, while the fluid sees a flat wall. At the aft end, the solid again has no applied load, while the fluid has an outflow boundary condition. The open-ended cylindrical case is taken to be rigid.

The calculation in Figure 5 was done on 8 processors and ran to 20 ms with a burn rate enhanced by a factor of 100 from the real propellant in order to test our regression capability in a reasonably short run time. As the pressure increases along the inner wall of the propellant, it bulges out at the forward and aft ends. Despite the deformation and high rate of regression, the solids and fluids meshes remain closely matched at the interface, as they should. We have also used Rocsolid to perform several calculations with regressing boundaries and verified that the numerical results closely match the analytical solution.

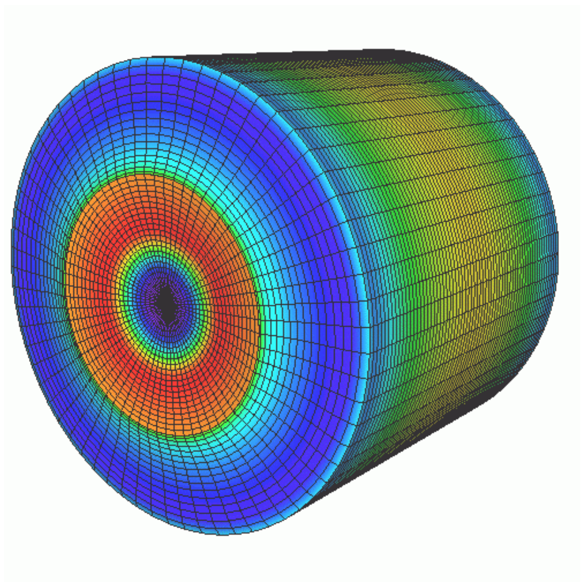

Figure 5. ALE Scalability Test Problem

To measure scalability, we increase the number of elements in the problem in proportion to the number of processors on which the simulation runs by adding additional cylindrical sections between the two ends. If scalability were perfect, the wall clock time would be the same for any number of processors, since each processor does the same amount of computational work and communication (expect at the ends, where no data needs to be communicated). Figure 6 shows the wall clock run times for one predictor-corrector cycle using 1, 8, 16, 32, 64, and 128 processors on ASCI White, a new IBM SP computer at Lawrence Livermore National Laboratory. Two data points are shown for ASCI Blue Pacific, and older SP. Because the ratio of system to fluids time steps is only 10 and the number of solid elements is relatively large in this test, the run time is dominated by the BiCGSTAB solver in Rocsolid (70 percent of the run time). For the range of problem sizes studied (8700 to over 1 M elements in the solid), the number of BiCGSTAB iterations is nearly constant, and scalability is very good.

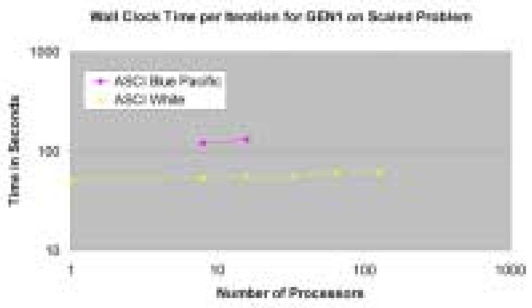

Figure 6. ALE Scalability Performance Data

Rocburn consumed 22 percent of the run time because of the relatively large burning surface area in this problem. Rocflo used just 8 percent, and Rocface required a very small amount of time for mesh association and interpolation.

Tactical Motor

We are currently simulating Tactical Solid Rocket Motor Number 13[24] (see Figure 7) using the GEN1 v2.0 package running on 128 processors on ASCI White. There is an empty cylindrical chamber at the head end of this rocket for instrumentation. At the early physical time of Figure 7, hot gas flows both aft out the nozzle and forward into the empty chamber (see also Figure 3 for a visualization of the velocity field and stress in the propellant). We are using Rocburn's dynamic burn rate to compute the pressure overshoot shortly after ignition. For this problem, we are using roughly 400,000 fluid cells and 200,000 solid elements. We find that when we take 100 fluids time steps for each system time step, Rocflo uses 63 percent of the wall clock time for a typical predictor-corrector iteration, Rocsolid uses 35 percent, and Rocburn uses only a fraction of a percent. Thus, the more expensive BiCGSTAB solver does not dominate the run time in this calculation as it did in the ALE scalability test problem.

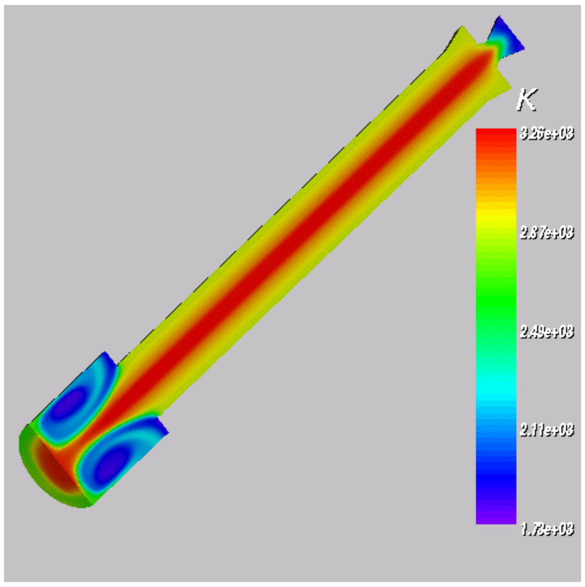

Figure 7. Temperature in Tactical Motor # 13

One difficulty that arises in our Motor 13 simulation is that the pressure inside the rocket soon exceeds the nominal Young's modulus of the propellant (36 Mpa). Since we have assumed for simplicity in this preliminary calculation that the propellant is a linear elastic material, we encounter far larger deformations than would be present in the real motor. Since the real propellant consists of aluminum and ammonium perchlorate particles in a much softer binder, it should be much stiffer under compression than it is under tension. Unfortunately, the Young's modulus we adopted was determined by studying propellants under tension. Although we are adding large deformation and viscoelastic material property capabilities to Rocsolid, we still require better material models of the propellant to ensure that the deformation we calculate is close to the deformation in the real rocket.

Our intention is to follow the regression of the propellant in Motor 13 until our meshes become too distorted to produce an accurate numerical solution. Reaching the blow-down phase will require the advanced remeshing capabilities under development for our GEN2 rocket simulation package. As a test of the mesh motion capabilities of both Rocflo and Rocsolid in GEN1 v2.0, we are running a somewhat artificial model problem with the same geometry as Motor 13, but with a power law burn rate that is enhanced from its nominal value by a factor of 1000. To keep the mass flux, and therefore the pressure history, roughly the same as the real Motor 13, we also reduced the propellant density by a corresponding factor.

Conclusions

CSAR has completed development of GEN1 version 2.0, a software package for detailed, 3-D, numerical simulation of solid propellant rockets. It solves the fully coupled fluid-structure interaction problem at solid surfaces, including a careful mass and momentum conserving treatment of regressing combustion interfaces. The application is scalable to large parallel supercomputers.

GEN1 is being applied to model a variety of rocket problems, including tactical motors in which the pressure overshoot that occurs shortly after ignition may be captured by the dynamic burn rate model included in the code.

References

[1] Dick, W. A., Heath, M. T., and Fiedler, R. A., 2001, "Integrated 3-D Simulations of Solid Propellant Rockets," AIAA Paper 2001-3949.
[2] Heath, M. T., Fiedler, R. A., and Dick, W. A., 2000, "Simulating Solid Propellant Rockets at CSAR," AIAA Paper 2000-3455.
[3] Parsons, I. D., Alavilli, P., Namazifard, A., Acharya, A., Jiao, X., and Fiedler, R., 2000, "Coupled Simulations of Solid Rocket Motors", AIAA Paper 2000-3456.
[4] Thomas, P. D., and Lombard, C. K., 1979, "Geometric Conservation Law and its Application to Flow Computations on Moving Grids". AIAA J., 17, 1030-1037.
[5] Jameson, A., Schmidt, W., and Turkel, E., 1981, "Numerical Solutions of the Euler Equations by Finite-Volume Methods Using Runge-Kutta Time-Stepping Schemes",. AIAA Paper 81-1259.
[6] Roe, P. L., 1981,. "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", Journal of Computational Physics, 43, 357-372.
[7] Yee, H. C., 1987, "Construction of Explicit and Implicit Symmetric TVD Schemes and Their Applications" Journal of Computational Physics, 68, 151-179.
[8] Pointwise, Inc., Fort Worth, TX, http://www.pointwise.com.
[9] Alavilli, P., Tafti, D., and Najjar, F. M., 2000, "The Development of an Advanced Solid-Rocket Flow Simulation Program ROCFLO", AIAA Paper 2000-0824.

[10] Ferry J., and Balachandar, S., 2001, “Multiphase Flow Research and Implementation at CSAR”, AIAA Paper 2001-3951.

[11] Bathe, K. J., 1996. Finite element procedures. Prentice-Hall.

[12] Parsons, I. D., 1997, “Parallel Adaptive Multigrid Methods for Elasticity, Plasticity and Eigenvalue Problems”, in Parallel Solution Methods in Computational Mechanics, M. Papadrakakis, editor, Wiley, 143-180.

[13] Namazifard, A., Parsons, I. D., Acharya, A., Taciroglu, E. , and Hales, J., 2000, “Parallel Structural Analysis of Solid Rocket Motors”, AIAA Paper 2000-3456.

[14] XYZ Scientific Applications, Inc., Livermore, CA, http://www.truegrid.com.

[15] Metis; http://www-users.cs.umn.edu/~karypis/metis/metis.html.

[16] Novozhilov, B. V., 1973, “Nonstationary Combustion of Solid Propellants”, Nauka, Moscow, (English translation available from NTIS, AD-767 945.

[17] Novozhilov, B. V., 1992, "Theory of Nonsteady Burning and Combustion Stability of Solid Propellants by the Zeldovich-Novozhilov Method", Nonsteady Burning and Combustion Stability of Solid Propellants, edited by L. De Luca, E. W. Price, and M. Summerfield, Vol. 143, Progress in Astronautics and Aeronautics, AIAA, New York, Chapter 15, pp. 601-641.

[18] Son, S. F., and Brewster, M. Q., 1993, "Linear Burning Rate Dynamics of Solids Subjected to Pressure or External Radiant Flux Oscillations," Journal of Propulsion and Power, Vol. 9, No. 2, pp. 222-232.

[19] Brewster, M. Q., M. J. Ward, and S. F. Son, 2000, "Simplified Combustion Modeling of Double Base Propellant: Gas Phase Chain Reaction Vs. Thermal Decomposition," Combustion Science and Technology, 154, 1-30.

[20] Tang, K. C. and M. Q. Brewster, 2001, "Dynamic Combustion of AP Composite Propellants: Ignition Pressure Spike," AIAA 2001-4502.

[21] Tang, K. C. and M. Q. Brewster, 2001, "Nonlinear Dynamic Combustion in Solid Rockets: L*-Effects," AIAA 2000-3572 and Journal of Propulsion and Power, Vol. 14, No. 4, 2001.

[22] Gossant, B., 1993, "Solid Propellant Combustion and Internal Ballistics of Motors," Solid Rocket Propulsion Technology, edited by A. Davenas, Pergamon Press, New York, Chapter 4.

[23] Blomshield, F. S., Crump, J. E., Mathes, H. B., Stalnaker, R. A., and Beckstead, M. W., 1997, "Stability Testing of Full-Scale Tactical Motors," Journal of Propulsion and Power, Vol. 13, No. 3, May-June, pp. 349-355.

[24] Blomshield, F. S., Crump, J. E., Mathes, H. B., and Beckstead, M. W., 1996, "Stability Testing and Pulsing of Full-Scale Tactical Motors", NAWCWPNS Technical Publication 8060.

[25] X. Jiao, M. T. Heath, and H. Edelsbrunner, 1999, “Mesh Association: Formulation and Algorithms,” Proceedings of the 8th International Meshing Roundtable, Tech. Report 99-2288, Sandia National Labs, Albuquerque, NM, pp. 75-82.

[26] Farhat, C., Lesoinne, M. and LeTallec, P., 1998. “Load and motion transfer algorithms for fluid/structure interaction problems with non-mathcing discrete interfaces”. Computer Methods in Applied Mechanics and Engineering, 157, 95-114.

[27] Jiao, X. and Heath, M. T., 2001, “Efficient and robust algorithm for computing common subdivision for nonmatching surface meshes”, Submitted.

[28] Jiao, X., 2001, “Efficient algorithms for moving interfaces in multicomponent simulations”, PhD thesis, University of Illinois at Urbana-Champaign. In preparation.

[29] Parsons, I. D. Alavilli, P., Namazifard, A., Jiao, X., Acharya, A., 2000, “Fluid-structure interaction through a non-material interface: simulations of solid rocket motors”. CDROM Proceedings of the 14th ASCE Engineering Mechanics Conference (EM2000), Austin Texas, May 2000.

[30] Rocpanda: http://cdr.cs.uiuc.edu/panda/rocpanda

[31] Lee, J., Winslett, M., Ma, X., &. Yu, S., 2001, “Tuning High-Performance Scientific Codes: The Use of Performance Models to Control Resource Usage During Data Migration and I/O”, To appear in Proceedings of the 15th ACM International Conference on Supercomputing, June 2001.

[32] Fiedler, R. A. and Norris, J. C., 2001, “Rocketeer User’s Guide”: http://www.csar.uiuc.edu/F_software/rocketeer

[33] Kitware, Inc.: http://www.kitware.com

[34] Schroeder, W. J, Martin, K. M., Avila, L. S. & Law, C. C., 2000, The VTK User’s Guide, Kitware, Inc.

[35] wxWindows: http://www.wxwindows.org/

[36] NCSA HDF version 4: http://hdf.ncsa.uiuc.edu/hdf4.html

[37] R. Fiedler and J. Norris, 2001, “Massively Parallel Visualization on Linux Clusters with Rocketeer Voyager”, Presented at Linux Clusters: the HPC Revolution, Urbana, IL, June 2001. http://www.csar.uiuc.edu/F_software/rocketeer/voyager

[38] ImageMagick: http://www.imagemagick.org