Fluid-Structure Interaction through a Non-Material Interface: Simulations of Solid Rocket Motors

I. D. Parsons, P. Alavilli, A. Namazifard, X. Jiao, A. Acharya

Center for the Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign

Abstract

We describe simulations of solid rocket motors that involve coupling between the core fluid flow, the structural response of the propellant and case and the combustion of the propellant. A predictor-corrector algorithm is employed to treat the fluid-structure interaction. The combustion rate of the propellant is coupled to the fluid flow via an empirical power law relationship. Our algorithm couples the physical processes involved using a partitioned approach, enabling us to use existing codes to perform the bulk of our simulations. We give special consideration to the jump conditions that hold at the fluid-structure-combustion interface. A large-scale simulation is described that demonstrates the capability of our code.

1. Introduction

Simulation of solid rocket motors presents challenges in many areas. A complete simulation requires consideration of different physics (e.g., fluid flow, structural deformation, fuel combustion, etc.), careful treatment of the interface between these zones and efficient parallel numerical algorithms. In this paper, we outline a preliminary implementation of a parallel code that couples the structural deformations experienced by the rocket fuel and casing with the fluid flow in the core of the rocket.

A code that will perform coupled multi-physics simulations can be designed using either a monolithic or a partitioned strategy. The monolithic approach requires that a single new code be developed by the various researchers who work in the different physics groups. Global iterations are performed that simultaneously update all of the variables in the different physics zones. The interface conditions are enforced as part of the global iterations. In contrast, the partitioned approach uses existing application codes that may have been developed by the different groups independently of any integration effort. Interface conditions are enforced by iteration between these different physics modules. Thus, the partitioned approach provides a relatively straightforward path for integration of the physics demanded by a complete simulation of a solid rocket motor.

The paper is constructed as follows. We first describe the three component codes: ROCFLO, ROCSOLID and ROCFACE. We then discuss the implementation of a partitioned algorithm that requires iteration to self-consistency between ROCSOLID and ROCFACE. We end the paper with a brief description of a large-scale demonstration simulation of the space shuttle solid rocket motor.

2. ROCFLO - The Fluids Solver

ROCFLO is a CFD code developed to simulate solid rocket booster core flow dynamics. Rocket flow problems exhibit singular features compared to other flow environments, primary being the fast propagating acoustic pulses in the chamber that could potentially cause instabilities and lead to rocket malfunction.

A three dimensional, structured, finite volume, cell-centered approach is adopted. In this framework complex geometries are handled through a multiblock approach. The multiblock approach also lends itself to parallel computing. The Navier-Stokes equations are solved on dynamic meshes whose boundaries adapt to conform to the propellant surface that deforms due to the loads imposed on it. Some details of the code are presented in this section; for further details consult [1].

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 [11] and Yee's symmetric TVD scheme [12]. A number of limiters, i.e., minmod, van Leer, van Albada and Superbee, have also been implemented. Yee's symmetric TVD scheme, along with one of the more diffusive limiters such as minmod, have been shown in the literature to perform very well in rocket chamber type simulations. Consequently this scheme is generally utilized in computations.

The flow equations are marched in time using an explicit multistage Runge-Kutta method. For the unsteady computations of the integrated rocket code, a two stage method is used as well as global time stepping, where the marching step size is limited by the smallest time step of all the cells in the computational domain.

ROCFLO is implemented in Fortran90. Fortran90 offers many features useful for parallel programming over Fortran77. Primary of these are the user defined data types. A computational block may be defined as an object and contains all data relevant to that block (node coordinates, face normals, volumes, solutions, parameters, etc.). This object model facilitates placement and migration of computational blocks on processors. The code was initially optimized for scalar performance both with respect to minimization of operations and also cache utilization. The code executes at about 80 Mflops on a single processor of a Origin2000. The code also scales very well on the Origin [9]. Further refinements are in progress.

3. ROCSOLID - The Structures Solver

ROCSOLID, the structural analysis code used in the coupled simulations, employs a finite element discretization of the problem domain using unstructured meshes. Dynamic problems are solved using the implicit Newmark time integrator [2]. The linear matrix equations encountered within the Newton iterations at each time step are solved using a scalable parallel multigrid solver [8]. The code is written in Fortran 90, and uses MPI to perform interprocessor communications.

Examination of the multigrid algorithm demonstrates that all of the operations can be performed independently on partitioned domains [10]. In particular, the main components of the algorithm are matrix-vector multiplications that can be efficiently implemented element-by-element. Interprocessor communications are only required during the 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; for example, the product $\boldsymbol{Kp}$, where $\boldsymbol{K}$ is a stiffness matrix and $\boldsymbol{p}$ is the search direction in conjugate gradient iteration, can be written as

$$\boldsymbol{Kp} = \sum_e \boldsymbol{K}^e \boldsymbol{p}^e = \sum_e \left(\int_{\Re^e} \boldsymbol{B}^T \boldsymbol{DB} dV \right) \boldsymbol{p}^e . \tag{1}$$

Computations can then proceed from right to left. The mesh is partitioned into a number of domains with equal numbers of elements and the product $\boldsymbol{Kp}$ is formed locally on each processor using the approach outlined in Equation (1); interprocessor communications are then performed using nonblocking MPI communications.

Multigrid methods require a hierarchy of increasingly finer meshes. We use *Truegrid* 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* to achieve perfect load balance between the processors. Uniform refinement of the coarsest mesh partitions produces the required partitions on all of the fine meshes. Thus, perfect element load balance is maintained through the mesh hierarchy, although the resulting communication pattern may not be optimum.

4. ROCFACE - The Interface Code

The interface code handles mesh association and data transfer between the fluid and solid meshes across multiple processors. The mesh association algorithm is used to determine the geometric relationship between the two non-matching meshes by locating the closest solid element for each fluid nodal point on the interface. Our algorithm traverses the fluid nodes and solid elements from neighbor to neighbor so that the closest elements can be located quickly [6]. We perform mesh association at every time step to track the moving interface. Since the geometric relationship changes rather slowly between the two meshes, from the second time step we search for the closest element of a fluid node starting from its closest element in the previous time step to speed up the search.

The result of the mesh association is then used for motion and load transfer between the fluid and solid meshes. It is essential to ensure conservation during data transfer; we are currently using the methods in [4]. Specifically, 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 mesh. For load transfer, the nodal forces for each solid node is evaluated as a weighed sum of the fluid nodal forces. The method uses the same set of coefficients for both load and motion transfer, and as a consequence, guarantees global conservation of energy [4].

Since the fluid and solid meshes are distributed across multiple processors, the interface code must handle distributed meshes. In our parallel implementation, we first compute the bounding boxes 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 geometric relationship 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,

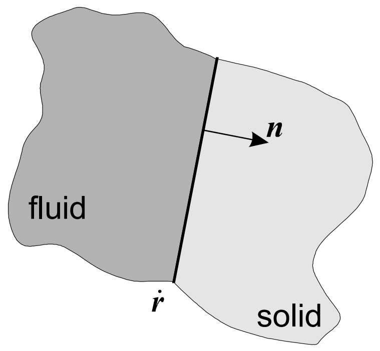

Figure 1: Interface geometry.

and are connected together to form a new mesh. The sequential mesh association algorithm is then applied on each processor in parallel. As the by-product of mesh association, a more accurate communication pattern is obtained which is used to communicate the physical quantities between processors to perform data transfer. The interface code is implemented in C++ using object-oriented techniques, employing the CGAL library (*www.cs.uu.nl/CGAL*) for the geometric primitives and the half-edge data structure for unstructured meshes.

5. GEN1 - A Coupled Multiphysics Code for Rocket Simulations

The three component codes described above form the basis of our rocket motor simulation tool. The key to successfully solving our coupled problem is to capture the interface physics correctly. Figure 1 shows the combustion interface between the solid and fluid domains; the interface moves with velocity $\dot{\boldsymbol{r}}$, and has an outward unit normal $\boldsymbol{n}$ measured positive into the solid domain. Consideration of conservation of mass and linear momentum requires that

$$\rho_s\left(\dot{\boldsymbol{r}}.\boldsymbol{n}-\boldsymbol{v}_s.\boldsymbol{n}\right)=\rho_f\left(\dot{\boldsymbol{r}}.\boldsymbol{n}-\boldsymbol{v}_f.\boldsymbol{n}\right)=m \tag{2}$$

and

$$m\left(\boldsymbol{v}_f-\boldsymbol{v}_s\right)+\left(\boldsymbol{t}_s^{(\boldsymbol{n})}-\boldsymbol{t}_f^{(\boldsymbol{n})}\right)=\boldsymbol{0} \tag{3}$$

respectively. Subscripts f and s denote the fluid and solid regions, respectively, ρ denotes density, m denotes the mass transfer into the fluid, $\boldsymbol{v}$ denotes velocity and $\boldsymbol{t}^{(\boldsymbol{n})}$ denotes tractions measured with respect to $\boldsymbol{n}$. In order to model solid rockets, we make some simplifying assumptions regarding the motion of the fluid-solid interface.

5.1 Stationary Interface

We first consider a stationary interface, where the interface is constrained to move with the solid, i.e., the interface regression rate, $\dot{\boldsymbol{r}}$, is equal to the solid velocity, $\boldsymbol{v}_s$. This assumption allows us to model the initial burn of the motor (e.g., $t<1$ sec), during which the regression of the interface through the solid is small. In order to model the effect of the mass transfer from the propellant to the core flow, we allow the interface to transfer mass at a rate per unit area m, where

$$m = \rho_s a p_f^n , \tag{4}$$

which is a standard combustion law. Consideration of the balance of mass and linear momentum produces

$$\boldsymbol{v}_f . \boldsymbol{n} = -\frac{m}{\rho_f} + \boldsymbol{v}_s . \boldsymbol{n} \tag{5}$$

and

$$\boldsymbol{t}_s = p_f \boldsymbol{n} - m \boldsymbol{v}_f , \tag{6}$$

respectively.

5.2 Moving Interface

In a simulation of the full rocket burn, we must consider a moving interface. Here, the interface will move through the solid according to the chosen combustion law. Following the standard combustion law used for the stationary interface, we have

$$\left(\dot{\boldsymbol{r}} - \boldsymbol{v}_s \right) . \boldsymbol{n} = a p_f^n . \tag{7}$$

The no-slip fluid boundary condition requires that

$$\boldsymbol{v}_f^t = \boldsymbol{v}_s - \left(\boldsymbol{v}_s . \boldsymbol{n} \right) \boldsymbol{n} \tag{8}$$

and balance of mass and linear momentum require

$$\boldsymbol{v}_f . \boldsymbol{n} = -\frac{\rho_s a p^n}{\rho_f} + \left(a p^n + \boldsymbol{v}_s . \boldsymbol{n} \right) \tag{9}$$

and

$$\boldsymbol{t}_s^{(-\boldsymbol{n})} = -\boldsymbol{t}_f^{(\boldsymbol{n})} + \rho_s a p_f^n \left(\boldsymbol{v}_s - \boldsymbol{v}_f \right) , \tag{10}$$

respectively.

5.3 Predictor-Corrector Coupling Algorithms

The interface conditions presented above are enforced using predictor-corrector cycles that iterate between the fluid and solid discretizations until self-consistency is obtained within a global time step. The algorithms follow methods described in the aeroelasticity literature [3,7], and are outlined in Figures 2 and 3 for the stationary and moving interfaces, respectively. To advance the solution from time t_n to t_{n+1} by advancing over a global time step Δt_n, we first advance the fluid solution a number of explicit time steps (e.g., 10). This produces estimates of the fluid interface pressure and velocity at time t_{n+1}, $p_f^{t_{n+1}}$, $\boldsymbol{v}_f^{t_{n+1}}$, respectively. Using Equations (6) and (10), the tractions acting on the solid at the interface, $\boldsymbol{t}_s^{t_{n+1}}$, are computed. These tractions are used as applied loads over an implicit solid time step ($\Delta t_s = \Delta t_n$). For coupling purposes, in the case of the stationary interface the solids calculation produces estimates of the interface position

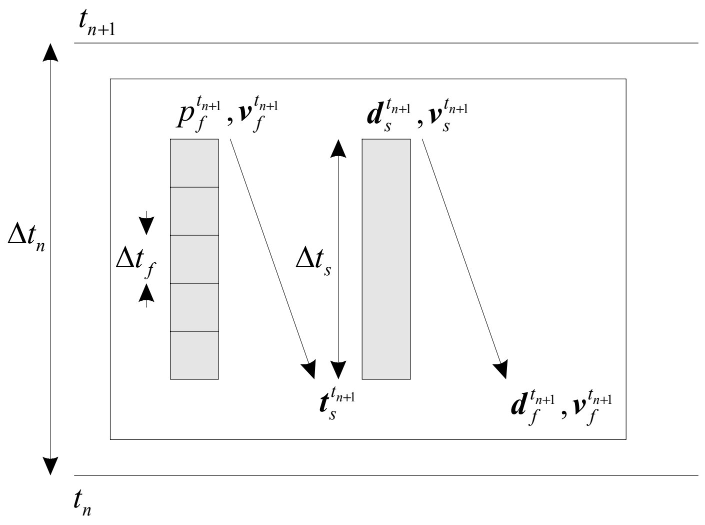
t_{n+1}
$p_f^{t_{n+1}}, \boldsymbol{v}_f^{t_{n+1}}$
$\boldsymbol{d}_s^{t_{n+1}}, \boldsymbol{v}_s^{t_{n+1}}$
Δt_n
Δt_f
Δt_s
$\boldsymbol{t}_s^{t_{n+1}}$
$\boldsymbol{d}_f^{t_{n+1}}, \boldsymbol{v}_f^{t_{n+1}}$
t_n

Figure 2: Stationary interface predictor-corrector coupling algorithm.

and velocity at t_{n+1}, $\boldsymbol{d}_s^{t_{n+1}}$ and $\boldsymbol{v}_s^{t_{n+1}}$; in the case of the regressing interface, the solid calculation generates the solid velocity at the interface $\boldsymbol{v}_s^{t_{n+1}}$. Using Equations (5) and (9), these quantities provide the fluid boundary conditions that can be used to drive the explicit fluids simulation, if necessary. In the case of the stationary interface they also prescribe the motion of the moving boundary for the fluid calculation; otherwise Equation (7) is used. The procedures shown in Figures 2 and 3 are repeated until the interface quantities have converged. We examine the relative change of the nodal velocities, displacements and forces on the solids side of the interface. When these quantities are small, we assume that the interface has converged. The algorithms used in ROCFACE ensure that conservative operators are used to transfer quantities across the fluids-solids interface.

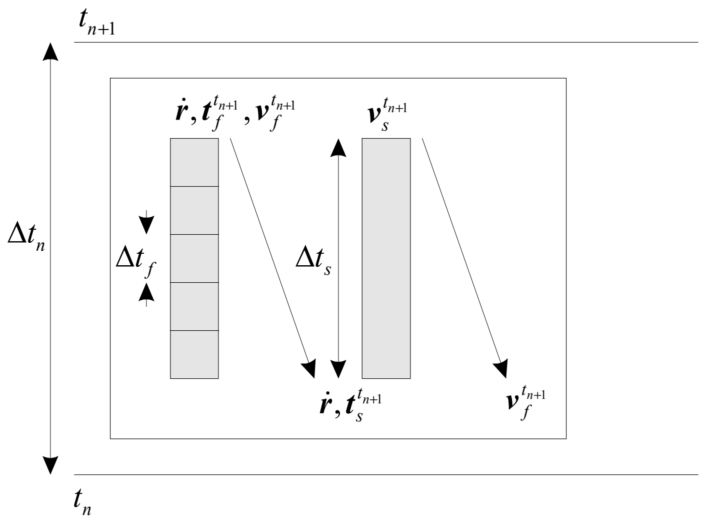
t_{n+1}
$\dot{\boldsymbol{r}}, \boldsymbol{t}_f^{t_{n+1}}, \boldsymbol{v}_f^{t_{n+1}}$
$\boldsymbol{v}_s^{t_{n+1}}$
Δt_n
Δt_f
Δt_s
$\dot{\boldsymbol{r}}, \boldsymbol{t}_s^{t_{n+1}}$
$\boldsymbol{v}_f^{t_{n+1}}$
t_n

Figure 3: Moving interface predictor-corrector coupling algorithm.

6. Space Shuttle Solid Rocket Motor Simulations

Our code is continuing to be developed to simulate a variety of solid rocket events. As a demonstration of our capabilities, we recently used a 256 processor Origin2000 in dedicated mode to simulate the first 0.1 secs of the firing of the space shuttle's solid rocket booster. ROCFLO used a discretization of 4,000,000 cells and ROCSOLID used a mesh of 270,000 elements (the fluid cells were typically ten times smaller in linear dimension that the solid elements). The fluids time step was 10^{-6} secs., resulting in a global and solids time step of 10^{-5} secs. Further details and visualizations produced from this simulation are available at *www.csar.uiuc.edu*.

References

[1] Alavilli, P. V. S., Tafti, D. and Najjar, F. 2000. The development of an advanced solid rocket flow simulation program ROCFLO. AIAA Paper 2000-0824, 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

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

[3] Farhat, C., Lesoinne, M. and Maman, N., 1995. Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution. International Journal for Numerical Methods in Fluids, 21, 807-835.

[4] 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.

[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] Jiao, X., Edelsbrunner, H. and Heath, M. T., 1999. Mesh association: formulation and algorithms. 8th International Meshing Roundtable, South Lake Tahoe, 75-82.

[7] Lohner, R. et al., 1995. Fluid-structure interaction using a loose coupling algorithm and adaptive unstructured grid. Computational Fluid Dynamics Review 1995, John Wiley, New York, 755-776.

[8] Namazifard, A. and Parsons, I. D., 1998. Parallel multigrid methods for structural mechanics using Fortran 90 and MPI. Poster presentation at Supercomputing 98.

[9] Parsons, I. D., Alavilli, P. V. S., Namazifard A., Hales, J. and Tafti, D., 1999. Coupled multi-physics simulations of solid rocket motors. Proceedings of the PDPTA'99 International Conference, 3101-3107.

[10] Parsons, I. D., 1997. Parallel adaptive multigrid methods for elasticity, plasticity and eigenvalue problems. Parallel Solution Methods in Computational Mechanics, Papadrakakis, M., editor, Wiley, 143-180.

[11] Roe, P. L., 1981. Approximate Riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics, 40, 263.

[12] Yee, H. C., 1987. Construction of explicit and implicit symmetric TVD schemes and their applications. Journal of Computational Physics, 68, 151-179.