Flow Simulation in Solid Rocket Motors Using Advanced CFD

Jiri Blazek*
Center for Simulation of Advanced Rockets (CSAR)
University of Illinois at Urbana-Champaign, Urbana, IL 61801

Abstract

This paper describes a recently developed flow solver intended for the simulation of solid rocket motors. It solves the integral form of the 3-D Euler/Navier-Stokes equations on moving and/or deforming grids. The grids are structured and can be composed of an arbitrary number of blocks. The object oriented design of the flow solver enables an easy addition of physical modules for the modeling of turbulence, particles, smoke, species and radiation. The solver contains a module to move the interior grid including the block boundaries according to the boundary deformation. Furthermore, the flow solver is able to exchange data with an exterior program through a set of standard interfaces. This feature is used to supply the flow solver with boundary conditions (e.g. related to the burning surface) as well as with the movement of the surface grid due to burn back and/or structural deformation. The numerical schemes utilized and the implementation are described in some detail. The accuracy of the new flow solver is demonstrated for three solid rocket motors.

1 Introduction

Nowadays, there is a growing demand for high-fidelity flow simulations in complex systems like in turbomachinery [1] or in solid rocket motors [2]. The goal is to obtain deep insight into the physical and chemical processes, which are difficult or impossible to measure. Also, it is required to simulate failure scenarios, which cannot be investigated experimentally. In order to achieve dependable predictions, it is necessary to base the numerical models on first principles rather than on correlations. Such complex simulations require not only powerful computer systems with up to several thousand processors, but also advanced solution methodologies and software design. The flow solver becomes one component of a coupled multidisciplinary system. In the case of solid rocket motors, for example, the fluids code has to communicate not only with physical modules for turbulence, particles, radiation, etc., but also with a structures code and a dynamic burn module.

This papers discusses in some detail the implementation of the new flow solver and the grid motion scheme for complex solid rocket motor simulations. It presents numerical results obtained for three different rocket motors as well as comparisons to experimental data.

2 Governing Equations

Equations being solved are the time-dependent Navier-Stokes equations in integral form on moving and/or deforming grid

$$\frac{\partial}{\partial t}\int_\Omega \vec{W}\, d\Omega + \oint_{\partial\Omega} (\vec{F}_c^M - \vec{F}_v)\, dA = \int_\Omega \vec{Q}\, d\Omega\,. \quad (1)$$

In Eq. (1), $\vec{W}$ denotes the vector of conserved variables, $\vec{F}_c$ the convective fluxes on moving grid, $\vec{F}_v$ the viscous fluxes, and Ω is the control volume with the surface $\partial\Omega$. Furthermore, dA represents a surface element of $\partial\Omega$ and $\vec{Q}$ stands for the source term. The convective fluxes $\vec{F}_c^M$ become on a dynamic grid

$$\vec{F}_c^M = \vec{F}_c - V_t\vec{W} \quad (2)$$

with $\vec{F}_c$ denoting the standard convective fluxes and V_t being the contravariant velocity of the face of the control volume. Hence,

$$V_t = n_x\frac{\partial x}{\partial t} + n_y\frac{\partial y}{\partial t} + n_z\frac{\partial z}{\partial t}\,, \quad (3)$$

*Senior Research Scientist, Senior Member AIAA.
E-mail: jblazek@uiuc.edu

where n_x, n_y, and n_z represent the Cartesian components of the outward facing unit normal vector of the surface $\partial\Omega$.

In order to avoid errors induced by a deformation of the control volumes, the Geometric Conservation Law (GCL) must be satisfied. The integral form of the GCL reads [3]

$$\frac{\partial}{\partial t}\int_\Omega d\Omega - \oint_{\partial\Omega} V_t \, dS = 0 \,. \tag{4}$$

Additional equations have to be solved along with Eqs. (1) and (4) in the case of a multi-phase flow and for the species concentrations. These will not be discussed in the present paper. In the simulations presented here, the evaluation of pressure, temperature and the speed of sound was based on the perfect gas assumption.

3 Spatial Discretization

The governing equations (1) are discretized and solved on 3-D, multi-block structured grids using a cell-centered finite-volume scheme. The convective fluxes $\vec{F}_c^M$ in Eq. (1) are approximated either by a central scheme with scalar artificial dissipation [4], or by the upwind Roe scheme [5]. The upwind scheme employs the MUSCL approach [6] with κ=1/3 and the limiter of Hemker and Koren [7], which results in slightly higher than 2nd-order accuracy on smooth grids. It is planned to implement the MAPS (Mach Number-Based Advection Pressure Splitting) scheme [8], which would be more suitable for chemically reacting flows.

The viscous fluxes $\vec{F}_v$ in Eq. (1) are centrally discretized in the standard way. Gradients of the velocity components and of the temperature at the cell faces are obtained using Green's theorem and an auxiliary control volume. A variety of LES schemes was recently added to the solver for the simulation of turbulent flows.

4 Temporal Discretization

The discretized governing equations (1) can be written in the form

$$\frac{d}{dt}(\Omega\vec{W}) + \vec{R} = 0 \,, \tag{5}$$

where the residual $\vec{R}$ is an approximation of the integral over the fluxes and the source term in Eq. (1). The set of ordinary differential equations (5) is solved in a time accurate fashion using either the classical 4-stage Runge-Kutta scheme in low-storage formulation or an implicit dual-time stepping scheme. In the later case, the time derivative in Eq. (5) is approximated by an implicit backwards difference formula. A 2nd-order accurate, A-stable scheme is given by [9]

$$\begin{aligned}\frac{3\Omega^{n+1}}{2\Delta t}\vec{W}^{n+1} - \frac{2\Omega^n}{\Delta t}\vec{W}^n + \frac{\Omega^{n-1}}{2\Delta t}\vec{W}^{n-1} \\ + \vec{R}(\vec{W}^{n+1}) = 0\end{aligned} \tag{6}$$

with n being the time level, i.e. $t = t_0 + n\Delta t$. A 3rd-order accurate implicit scheme is also available in the code. It reads [10]

$$\begin{aligned}\frac{11\Omega^{n+1}}{2\Delta t}\vec{W}^{n+1} - \frac{9\Omega^n}{\Delta t}\vec{W}^n + \frac{9\Omega^{n-1}}{2\Delta t}\vec{W}^{n-1} \\ - \frac{\Omega^{n-2}}{\Delta t}\vec{W}^{n-2} + \vec{R}(\vec{W}^{n+1}) = 0 \,.\end{aligned} \tag{7}$$

Equation (6) or (7) is solved by driving a modified system (5), i.e.

$$\frac{d}{d\tau}(\Omega\vec{U}) + \vec{R}^*(\vec{U}) = 0 \tag{8}$$

to steady-state in the pseudo time τ (thus $\vec{R}^* = 0$). In Eq. (8), $\vec{U}$ is an approximation to $\vec{W}^{n+1}$ and $\vec{R}^*$ is defined by Eq. (6) or (7), respectively. Currently, the integration of Eq. (8) in pseudo time is accomplished by an explicit multistage scheme accelerated by local time stepping and implicit residual smoothing. However, it is intended to employ the implicit LU-SGS scheme [11], [12] and multigrid [13] instead. Even in its present form, the implicit scheme is able to reduce the CPU-time for the simulation of an inviscid flow in a typical solid rocket motor by a factor of 6 to 7 as compared to the explicit Runge-Kutta scheme.

5 Grid Motion

Due to the burn-back of the solid propellant and structural deformations, it is necessary to constantly adjust the interior grid. It is required that the grid motion works in parallel on all blocks since the complete grid cannot be kept in the memory of one processor. Parallel grid motion is also important for computational efficiency of the flow solver. Two different grid motion schemes are available in the code and will be described next.

The first algorithm initially moves the grid in each block separately by using the linear Transfinite Interpolation (TFI) method [14] with the blending functions of Soni [15]. TFI is employed to interpolate the deformation of the block boundaries onto the grid

from the previous time step. After this initial step, deformations are exchanged between the blocks. This is important in cases where a block shares only a point or a line with the moving surface, as the block 2 in Fig. 1. As the last step, grids are regenerated inside those blocks, whose boundaries were moved by adjacent blocks.

In cases with a large surface movement, it becomes necessary to deform and move grid blocks even far away from the surface. This situation is handled by a second algorithm. This scheme globally smoothes the grid and the block boundaries by solving a Laplace equation [16] for the displacements. Jacobi iteration is employed for this purpose since it can be easily parallelized. After the smoothing, the original spacing of the grid lines is mapped back onto the block boundaries. This way, no source terms are needed to control the grid spacing, which results in a faster and more stable algorithm. Grid inside the blocks is then regenerated using TFI as in the first method.

6 Implementation

The flow solver is written in Fortran 90 using an object oriented approach. User derived types are employed to encapsulate data of the core solver and of each physical module (like turbulence, particles, smoke, etc.). A set of standardized interfaces handles all interactions between the core solver and the modules, as well as between the modules themselves. This greatly simplifies the upgrade or replacement of a module. Whenever possible, generalized functions are provided (like for the calculation of face gradients), which can be utilized by any of the physical modules. In order to facilitate the use of different gas models, all calculations of dependent variables like pressure, temperature, speed of sound, or viscosity are encapsulated in a single module.

The flow solver is implemented as a library with two top-level functions for initialization and solution advancement. The library is linked to an external driver, which provides boundary conditions for all burning and/or deforming surfaces, and also controls the overall time-stepping procedure. Thus, there is no rocket specific code inside the flow solver. The idea is to encapsulate the three main tasks: flow solution, structural response and dynamic burning into separate libraries. The driver program is then responsible for the data exchange [2].

The flow solver allows the use of different physical or gas models in each block (e.g. LES – RANS). This feature is also important for the simulation of the internal together with the external flow past a rocket in flight. The solver is prepared for the treatment of non-matching grid interfaces. Furthermore, work is in progress on the coupling to an unstructured grid [17], [18]. Standard MPI is employed for the parallelization. In order to improve the parallel efficiency, communication is hidden behind computation. The performance for a scaled problem (constant work per processor) can be seen in Fig. 2 for up to 960 processors.

Physical boundary conditions and the data exchange between the blocks utilize the concept of dummy cells. A correct implementation of the viscous terms and of LES in a multi-block framework requires a proper treatment of dummy cells located at the corners and edges of the computational space. The donor block and cell for each edge or corner cell of a block are located by a search procedure working in the index space. The search is conducted only once at the beginning of a run, or if the grid topology changes. This approach avoids the usual multiple sends and receives when flow variables in the edge and corner cells are updated.

7 Simulation Results

The flow solver was initially validated using relatively simple test cases like the laminar flat plate or the nozzleless rocket motor [19], for which analytical solution or experimental data are readily available. Here, larger and more complex test cases are discussed. All simulations reported here were conducted for inviscid flow only.

7.1 Motor 13

The first case represents the NAWC Tactical Motor Number 13 [20], [21]. The motor is about 2 meters long and the propellant consists of a single cylindrical section, which is tapered at the rear end. There is a large room at the head end for instrumentation. The surface grid of the fluids domain is displayed in Fig. 3. The domain behind the nozzle, which can be partly seen in the plot, encloses the plume region. The simulation was started with all propellant ignited. Grid movement due to burn back was neglected. A comparison between the numerically obtained quasi-steady head-end pressure and measured data is given in Fig. 4. Two different grids were employed, a fine one with 414,208 cells and a coarse grid with 76,608 cells. Because of the instantaneous ignition and dynamic burning not being modeled, the simulation shows much faster pressurization and no spike. However, the quasi steady pressure agrees

for both grids reasonably well with the experimental data. A snapshot of the static temperature in mid-section is shown in Fig. 5. As it can also be seen in Fig. 6, the flow in the head end section is highly non-symmetrical. The flow pattern also changes in time. This behavior needs to be further investigated using a much finer grid and viscous flow.

7.2 Titan IV SRMU

This case represents the complete flow domain of the Titan IV SRMU. All relevant geometrical features were modeled, including the star-grain region, the slots between the three segments as well as the stress relieve grooves. The surface of the initial fluids domain can be seen in Fig. 7 together with the block boundaries. As in the case of Motor 13, the plume region was modeled as well. The time history of the head-end pressure is displayed in Fig. 8. It was again assumed that the whole propellant was initially ignited. Measurements and simulation of the head-end pressure presented in Ref. [22] indicate a value of about 10 MPa. The result of the current simulation is 9.5 MPa, which is in a reasonable agreement. The runtime for 1.6 seconds of physical time was about 20 hours on 512 processors (IBM-SP). The temperature field shortly after ignition is displayed in Fig. 9.

7.3 Attitude Control Motor

The last case represents a small (few inches long), fast burning solid rocket motor. The surface grid of the initial fluids domain is rendered in Fig. 10. The complete burn out was simulated in this case. The movement of the surface nodes due to burn back was approximated by Huygens' construction. A comparison of the head-end pressure between the present flow solver and the prediction by a proprietary analysis code (validated by experiments) is depicted in Fig. 11. The agreement is very favorable, in particular in the case of the Roe upwind discretization. The head end pressure exhibits significant oscillations in time due to acoustic disturbances. However, Fourier analysis of the pressure history has not yet been conducted. The thrust history, which is compared in Fig. 12, is also in a good agreement with the prediction. The accuracy of the present solver is likely to improve by the addition of dynamic effects on the burn rate during ignition, as well as by the incorporation of nozzle ablation into the simulation. Contours of static temperature in the mid-section of the rocket motor are displayed in Fig. 13.

8 Conclusions

A recently developed flow solver for complex 3-D simulations of solid rocket motors in was presented. The numerical approaches as well as the implementation were described in some detail. The flexible modular structure of the solver allows an easy coupling to additional physical modules. The flow solver is designed in such a way that it can exchange boundary values with codes from other disciplines, like for example from structural mechanics. The flow solver contains two different grid motion schemes. The first one is based on a block-wise application of TFI for the grid deformations. The second algorithm smoothes the grid globally by using a Laplacian operator. Comparisons of numerical results to experimental data or to other predictions were presented for three solid rocket motors. It was demonstrated that despite the lack of tuned correlations, the current flow solver is able to predict even a full burn out with good accuracy.

Acknowledgments

CSAR is supported by the U.S. Department of Energy through the University of California under subcontract B523819. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the U.S. Department of Energy, the National Nuclear Security Agency, or the University of California.

References

[1] Blazek, J.; Irmisch, S.; Haselbacher, A.: *Unstructured Mixed-Grid Navier-Stokes Solver for Turbomachinery Applications*. AIAA Paper 99-0664, 1999.

[2] Fiedler, R.A.; Breitenfeld, M.S.; Jiao, X.; Haselbacher, A.; Geubelle, P.; Guoy, D.; Brandyberry, M.: *Simulations of Slumping Propellant and Flexing Inhibitors in Solid Rocket Motors*. AIAA Paper 2002-4341, 2002.

[3] Thomas, P.D.; Lombard, C.K.: *Geometric Conservation Law and Its Application to Flow Computations on Moving Grids*. AIAA Journal, 17 (1979), pp. 1030-1037.

[4] Jameson, A.; Schmidt, W.; Turkel, E.: *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes*. AIAA Paper 81-1259, 1981.

[5] Roe, P.L.: *Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes*. J. Computational Physics, 43 (1981), pp. 357-372.

[6] Van Leer, B.: *Towards the Ultimate Conservative Difference Scheme V. A second Order Sequel to Godunov's method*. J. Computational Physics, 32 (1979), pp. 101-136.

[7] Hemker, P.W.; Koren, B.: *Multigrid, Defect Correction and Upwind Schemes for the Steady Navier-Stokes Equations*. Synopsis, HERMES Hypersonic Research Program Meeting, Stuttgart, Germany, 1987.

[8] Rossow, C.-C.: *A Flux Splitting Scheme for Compressible and Incompressible Flows*. AIAA Paper 99-3346, 1999.

[9] Jameson, A.: *Time Dependent Calculations Using Multigrid with Applications to Unsteady Flows Past Airfoils and Wings*. AIAA Paper 91-1596, 1991.

[10] Arad, E.; Martinelli, L.: *Large Eddy Simulation of Compressible Flow Using a Parallel, Multigrid Driven Algorithm*. AIAA Paper 96-2065, 1996.

[11] Yoon, S.; Jameson, A.: *LU Implicit Schemes with Multiple Grids for the Euler Equations*. AIAA Paper 86-0105, 1986; also AIAA Journal 7 (1987), pp. 929-935.

[12] Rieger, H.; Jameson, A.: *Solution of Steady 3-D Compressible Euler and Navier-Stokes Equations by an Implicit LU Scheme*. AIAA Paper 88-0619, 1988.

[13] Blazek, J.: *A Multigrid LU-SSOR Scheme for the Solution of Hypersonic Flow Problems*. AIAA Paper 94-0062, 1994.

[14] Gordon, W.N.; Hall, C.A.: *Construction of Curvilinear Coordinate Systems and Application to Mesh Generation*. Int. J. Num. Methods in Engineering, 7 (1973), pp. 461-477.

[15] Soni, B.K.: *Two- and Three-Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics*. AIAA Paper 85-1526, 1985.

[16] Thompson, J.F.; Soni, B.K.; Weatherill, N.P. (eds.): *Handbook of Grid Generation*. CRC Press, Boca Raton, 1999.

[17] Blazek, J.: *Conservative Coupling Algorithm for Structured-Unstructured Grid Interfaces*. AIAA Paper 2002-2974, 2002.

[18] Blazek, J.: *Comparison of Two Conservative Coupling Algorithms for Structured-Unstructured Grid Interfaces*. AIAA Paper 2003-3536, 2003.

[19] Traineau, J.-C.; Hervat, P.; Kuentzmann, P.: *Cold-Flow Simulation of a Two-Dimensional Nozzleless Solid Rocket Motor*. AIAA Paper 86-1447, 1986.

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

[21] Blomshield, F.S.; Crump, J.E.; Mathes, H.B.; Stalnaker, R.A.; Beckstead, M.W.: *Stability Testing of Full-Scale Tactical Motors*. J. Propulsion and Power, 13 (1997), pp. 349-355.

[22] Johnston, W.A.; Murdock, J.W.: *Flow-Structural Interaction Inside a Solid Rocket Motor During Ignition Transient*. J. Propulsion and Power, 11 (1995), pp. 998-1005.

Figures

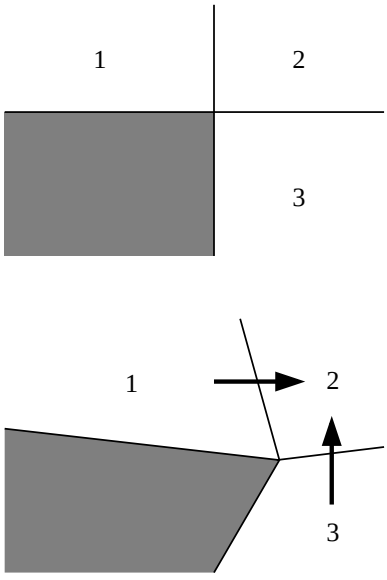

Figure 1: Adjustment of the boundaries of a block (2) which does not share a cell face with the deforming body (gray). Boundary movement is communicated through the blocks 1 and 3. Top: undeformed, bottom: deformed configuration.

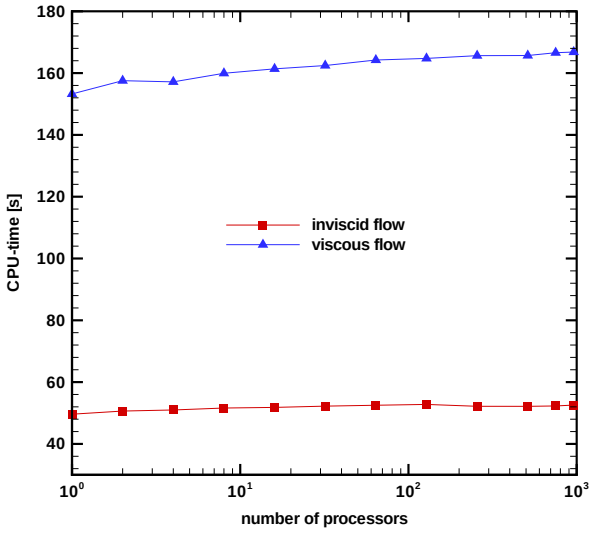

Figure 2: Parallel performance for a scaled problem with 20,000 cells per processor on ASCI White.

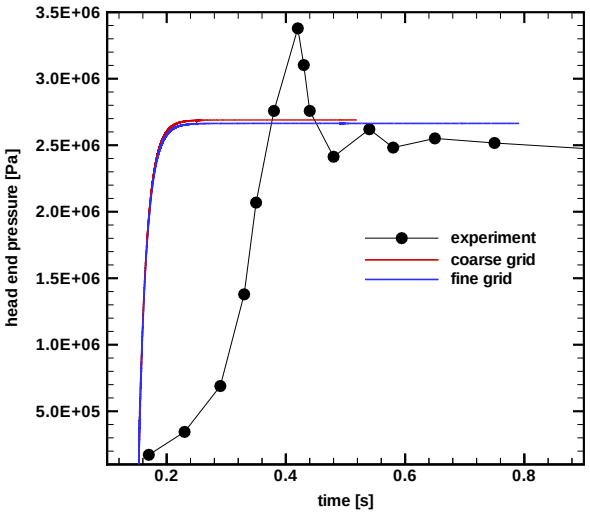

Figure 4: Motor 13; time history of the head-end pressure for two different grids.

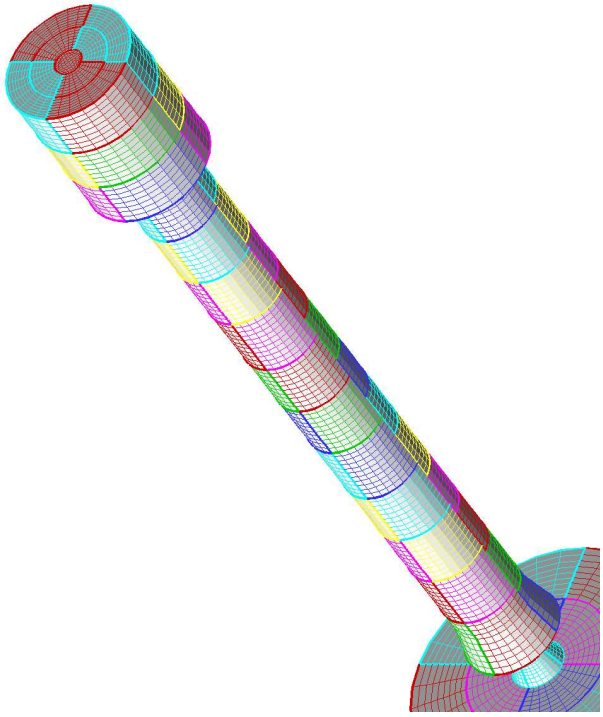

Figure 3: Motor 13; fine grid with 414,208 cells and 128 blocks.

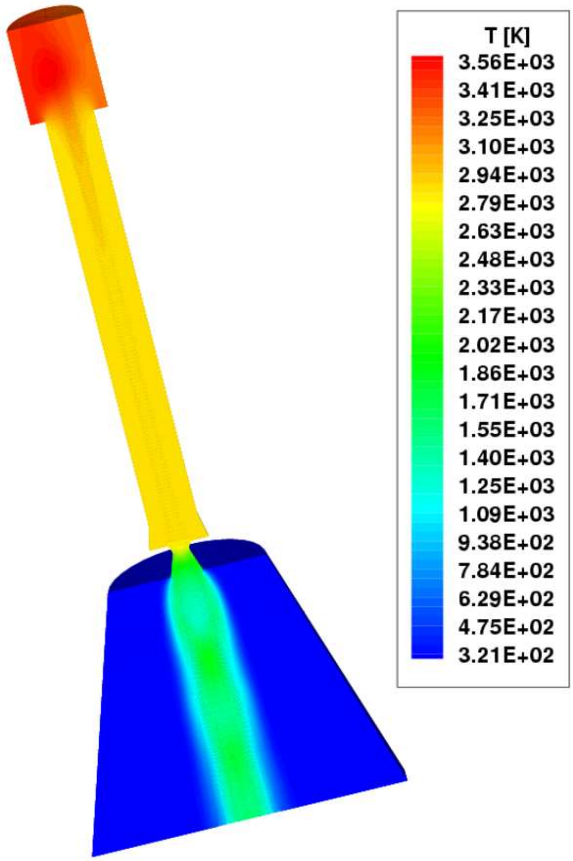

Figure 5: Motor 13; contours of the static temperature (in degree Kelvin) in the mid-section at $t = 0.1$s.

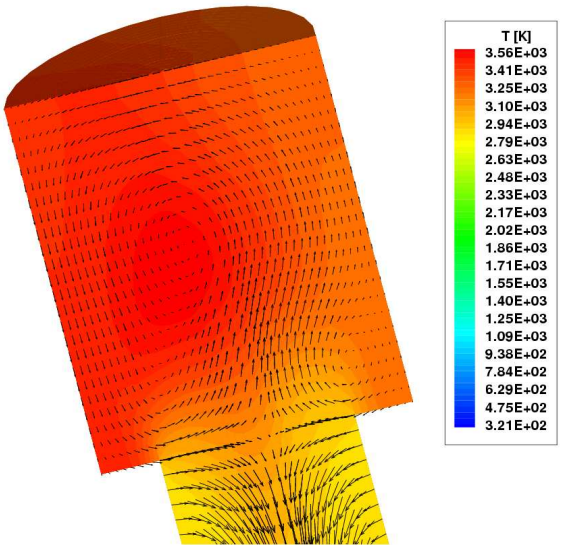

Figure 6: Motor 13; contours of the static temperature and the velocity vectors in the mid-section of the head end at $t = 0.1$s.

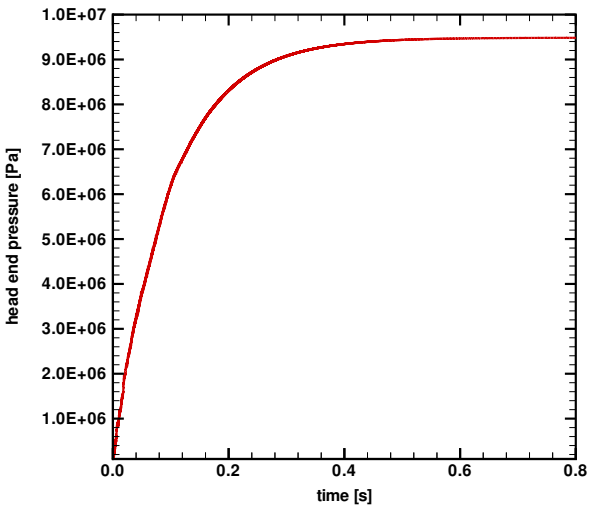

Figure 8: Titan IV; time history of the head-end pressure.

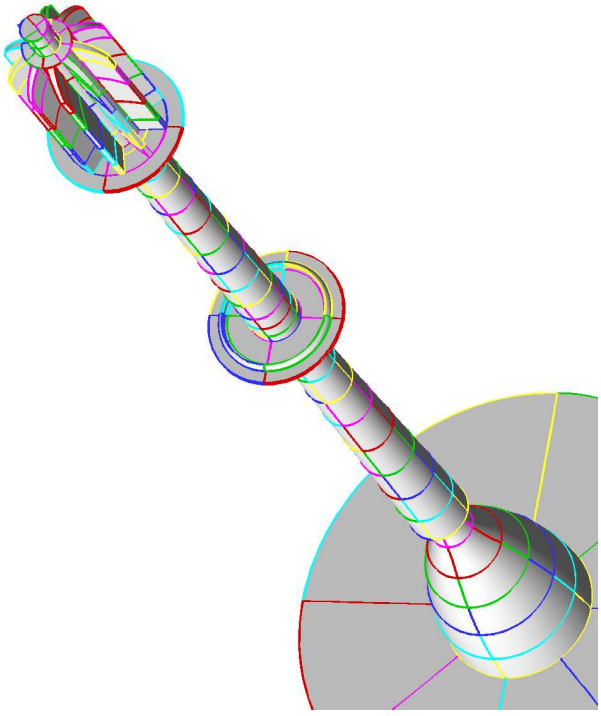

Figure 7: Titan IV; grid with 2,044,200 cells and 512 blocks (only the block boundaries are shown).

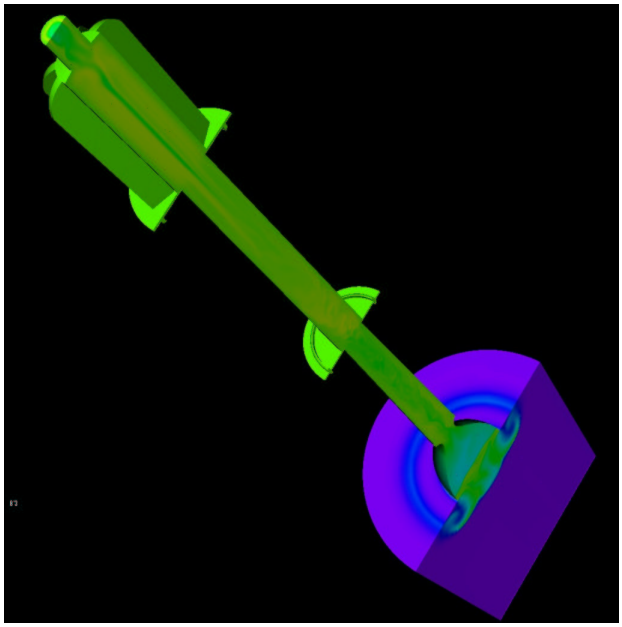

Figure 9: Titan IV; contours of the static temperature in the mid-section at $t = 30ms$.

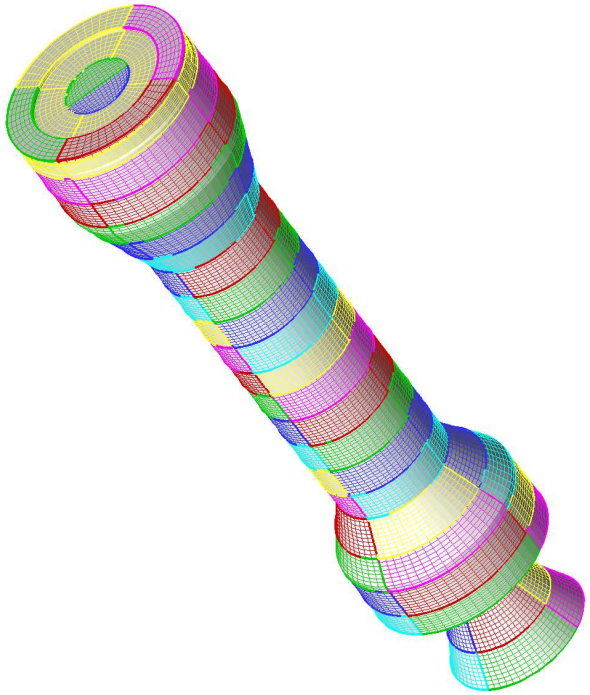

Figure 10: Attitude control motor; grid with 220,416 cells and 128 blocks.

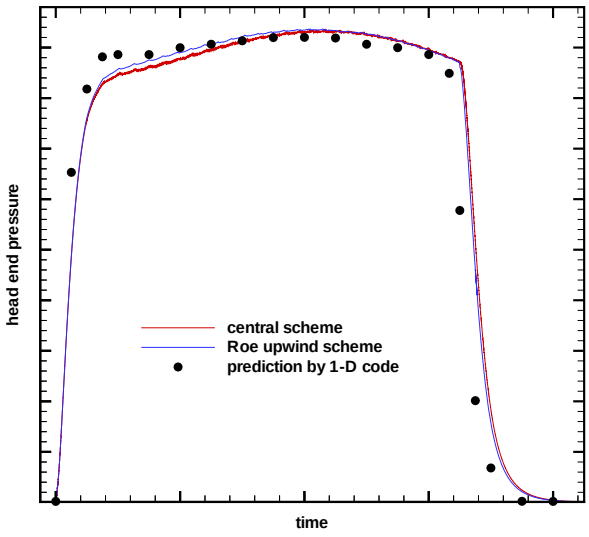

Figure 11: Attitude control motor; time history of the head-end pressure for two different discretizations.

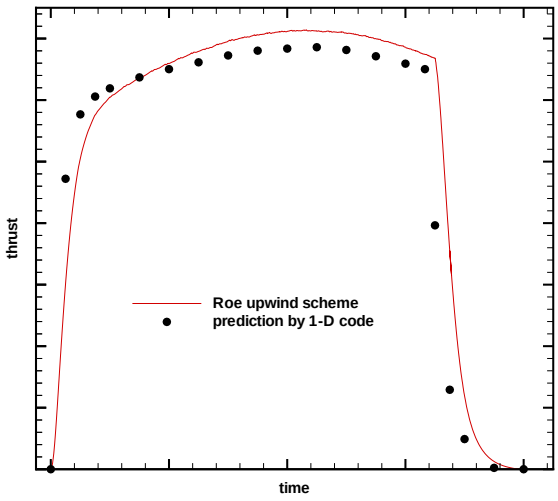

Figure 12: Attitude control motor; time history of the total thrust.

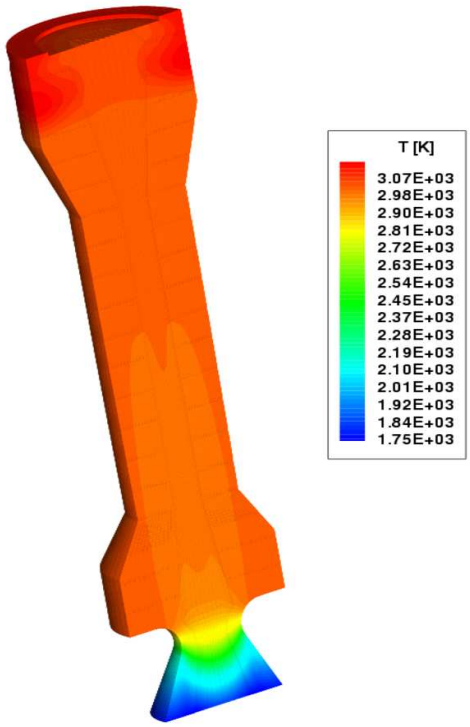

Figure 13: Attitude control motor; contours of static temperature (in degree Kelvin) in the mid-section at $t = 6.6$ms.