ADAPTIVE SIMULATION OF THE INTERNAL FLOW IN A ROCKET NOZZLE

Luciano Garelli, Gustavo Ríos Rodriguez, Rodrigo Paz and Mario Storti

Centro de Investigación de Métodos Computacionales (CIMEC)
Universidad Nacional del Litoral (UNL) - CONICET
Predio CONICET - Santa Fe, Colectora Ruta Nac.168 / Paraje El Pozo,
Santa Fe (S3000GLN), Argentina
gusadrr@santafe-conicet.gov.ar

***Abstract*— This work is a first step in the understanding of the interaction process between internal shock waves and the flow transition inside of a rocket nozzle that develops during the engine start-up phase or when the nozzle is operated at over-expanded conditions. In many cases, this transition in the flow pattern produces side loads in the nozzle due to an asymmetric pressure distribution on the wall, being harmful for the rocket's integrity. To understand this phenomenon, a numerical simulation is performed by solving the three-dimensional Euler equations on unstructured tetrahedral meshes. With this model the computational cost to solve the equations significantly increases, therefore parallel processing is required. Also, an unsteady h-adaptive refinement strategy is used jointly with a Streamline Upwind Petrov-Galerkin and a discontinuity capturing scheme, both to keep the size of the fluid flow problem bounded and to sharply resolve the shock wave pattern. The mesh adaptation strategy is introduced. Since its performance is a major concern in the solution of unsteady flow problems, some implementation issues about the data structure chosen to represent the mesh are discussed. Average pressure distributions computed at the wall and the axis of the nozzle for various pressure ratios are analyzed based on experimental and numerical results from other authors.**

***Keywords*— Rocket nozzle, shock wave transition, adaptive refinement, unstructured grids, mesh data structures.**

I. INTRODUCTION

Nozzles with high area ratio are used in the main space launchers (e.g. Space Shuttle Main Engine, Ariane 5). The nozzle contour is often designed according to the theory proposed by Rao (1996) that results in a TOP (Thrust Optimized Parabolic or Parabolic Bell Nozzle) nozzle, which has important advantages compared to other well known contours (i.e, Conical, Truncated Ideal and Thrust Optimized Contours). These advantages are smaller length, lower weight, as well as the reduction in energy losses in the expansion of the gases (Sutton and Biblarz, 2001; Oates, 1997; Mattingly and Ohain, 2006; Tuner, 2006). Also, in a TOP nozzle the internal shock that develops affects the flow properties at the wall, giving a slightly higher wall pressure at the nozzle exit which helps to avoid flow separation at sea-level conditions (Ostlund, 2002). A rocket nozzle must work in much varying conditions, ranging from sea level to orbital altitude. However an efficient operation is only reached at high altitude, so the nozzle operates under strongly over-expanded conditions during a large part of the flight envelope. The transition from an over-expanded condition to an ideal expansion produces a change in the flow pattern inside the nozzle, which in turns produces side loads due to a change in the pressure distribution on the wall. The aim of this work is to numerically solve this transition in the internal flow pattern for the VOLVO S1 subscale TOP nozzle on a 3-D geometry with the aid of a mesh adaptation strategy. Considering that future research involves the numerical simulation of side loads due to asymmetric pressure evolution on the nozzle wall, a fluid structure interaction algorithm (Garelli et al., 2011, 2010) will be used. The flow that develops inside the nozzle is perfectly suited for being adaptively solved because flow features of interest develop in very thin regions compared to some characteristic length of the nozzle. Adaptation of the mesh allows to reduce the computational effort required to solve the flow problem since elements are introduced only when and where they are needed. In this work, an unsteady h-refinement adaptation algorithm for unstructured finite element meshes is adopted. Since unsteady problems require to refine and unrefine the mesh a great number of times in order to follow discontinuities in their travel throughout the computational domain, the adaptation of the mesh should demand a small fraction of the overall solution time. To this end, the un-

derlying data structure chosen for the mesh representation and its corresponding implementation must be specially suited to allow efficient refinement and unrefinement. In this sense, we make extensive use of the containers and algorithms of the Boost (Boo, 1998-2012; Schäling, 2011) and Standard Template Library (Sil, 1993-2012; Schildt, 1998) for the C++ language. Also, the adaptation scheme ought to minimize the geometrical quality degradation of the mesh for convergence reasons (Shewchuck, 2002). To address these issues an h-refinement strategy for linear tetrahedra and hexaedra, uniquely based on the regular $1:8$ element subdivision is considered. No transition elements are used to match zones with different levels of refinement so that hanging nodes on edges and faces arise and the refined mesh is non-conforming. Extension to 3-D meshes of the well known 1-irregular vertex refinement constraint is used to ensure that neighbour elements in the mesh have similar size. This avoids the effort of considering and managing a great number of transition templates (Staten, 1996; Löhner and Baum, 1992; Remacle et al., 2002) for eliminating the hanging nodes and keeps bounded the quality degradation of the mesh (Ríos Rodriguez et al., 2005; Rios Rodriguez et al., 2009). As a consequence, the refinement algorithm used in this work results simple and scales almost linearly with the number of refined elements (Ríos Rodriguez et al., 2011).

The solution procedure is partially parallelized, which means that the adaptation of the mesh is sequentially performed while the solution of the flow equations is computed in parallel on a Beowulf cluster (Storti, 2005-2010) using the PETSc-FEM software (Storti et al., 1999-2010; Sonzogni et al., 2002). This latter is a multi-physics Object Oriented Programming code which uses both a finite element SUPG formulation to stabilize the advective terms of the equations and shock capturing terms for the treatment of non-linear instabilities in the neighbourhood of shocks (Brooks and Hughes, 1980, 1982a; Hughes and Mallet, 1986a,b; Tezduyar and Senga, 2006). Since continuous finite element functions are considered, constraints to the solution field at irregular vertices of the mesh must be applied. A Mach number-based gradient indicator is used to tag the cells of the mesh that need to be refined or coarsened. The adaptation of the mesh and the solution computation are coupled through an interface which automates the procedure. In this manner, the boundary conditions for the problem are specified at the starting mesh and are automatically updated later. Also, an interpolated state on the new adapted mesh is given in order to resume the flow computation.

Finally, pressure distributions for different nozzle pressure ratios are presented at the wall and the axis of the nozzle. They are analyzed based on numerical and experimental results from other authors (Ostlund, 2002). The flow pattern transition is apparent.

In summary, this work focuses in the numerical solution of the three-dimensional unsteady flow inside of a rocket nozzle by using a mesh refinement technique in conjunction with a parallel finite element flow solver. Although the use of mesh adaptation techniques may be considered as a standard procedure in engineering calculations, solving unsteady flows such as those presented in this work is still a challenging problem.

II. MESH ADAPTATION

A. Data structure for mesh representation

The data structure chosen to represent the mesh is a key aspect to be considered in the design of the adaptation code, since it has a great influence on both the execution time and the memory consumed. The code used in this work is written in C++ language and makes extensive use of the containers, iterators and algorithms of the Standard Template Library and the Boost programming libraries. This last one is specifically designed to work well with the C++ Standard Library.

The mesh is represented by geometrical entities of different dimension and their corresponding adjacencies. Each entity is uniquely identified by an iD number. We naturally define the **ele**, **face** and **edge** C++ classes to represent the corresponding mesh entities. Instances of these classes are stored in STL vectors, since this provides random access to each entity in the container, which is a requirement of the refinement / coarsening algorithm. Coordinates of the vertices are stored in a 2-D dynamic-size array from the Boost Multidimensional array library. Boost multi arrays are chosen because they are a more efficient way to represent n-dimensional dynamic-size arrays than an equivalent implementation provided by nested vectors of the STL library and they provide a much cleaner and less error prone interface than native C++ arrays (Boo, 1998-2012).

For the mesh representation we choose the following set of *downward* adjacencies: **faces-of-element**, **edges-of-element**, **vertices-of-element**, **edges-of-face** and **vertices-of-edge**. Even if it is not strictly necessary to store the edges-of-element set, we do because of the great number of times that we need to retrieve this information and the implicit cost that requires the second order (indirect) access of retrieving the faces-of-an-element, then the edges-of-a-face and finally computing the union of this final set of edges. Similar reasons lead us to manage the vertices-of-element adjacencies. This conclusion is based on concepts and analysis developed in (Remacle and Shephard, 2003). Since the number of downward adjacencies for an entity of a given dimension is known beforehand (i.e. a tetrahedra has four faces, six edges and four vertices), fixed size arrays can be used to store them. Boost bidimensional array containers are used to implement the downward adjacencies for all the entities of the same dimensionality. This allows random access and enables to preserve the *local order* given by

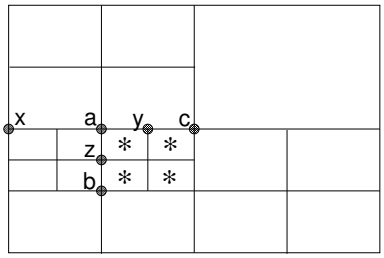
downward entity templates. Templates allow to manage the concept of "orientation" to ensure uniqueness of the entity representation, that is they describe local relationships between downward entities of a same entity type.

On the other hand, *upward* adjacencies are also required in different stages of the algorithm. In this case, the number of entities of a lower dimension that "share" an entity is not known beforehand. This means that it is not convenient to use fixed-size containers. Also, the order of the upward adjacencies has no importance, so we use unordered multimap containers (Boo, 1998-2012). These are hashed-type multiple associative containers which allow to access data with constant complexity on average. The upward adjacencies for a given entity are retrieved by iterating through the range in the hashed container whose keys are the same as the entity iD.

For unsteady problems the adaptation of the mesh requires to insert as well as to erase entities in the data structure. The entities that represent the current mesh are defined as "active" and their iDs are stored in unordered (hashed) sets. If an entity is refined, it is kept in the data structure, but its iD is replaced by those of its children in the set of active entities. In this case it is said that the refined entity is "deactivated". If the child entities are later unrefined, their iDs are erased from the active entities set and the parent entity iD is "activated" again by inserting its iD in the set. Unordered sets are used because it is required to erase an iD at any position in the container while the order in which the iDs are stored is not important. Unordered sets allow to insert and erase elements with constant complexity on average.

Since vectors are used to store the entities, it is not efficient to explicitly erase them from the container because this takes linear time on the size of the vector unless this is done at the end (which is not the common situation). We consider that if an entity is unrefined, its iD is inserted in a list of "free iDs" for that kind of entities. This means the iD is available to be reused by a new entity of the same kind (dimension). If new entities of the same dimension are created later during refinement, their iDs are taken from this free iDs list until it is left empty. If the number of required iDs is more than the ones affordable in the free iDs list, the remaining iDs are generated in sequence, starting from the biggest one.

A data object of the element class stores the iDs of the elements that arise in the mesh because of its refinement (children), the iD of the element from which derives (parent), the refinement level to which belongs, the iDs of its vertices (following a local order or orientation) and a property label that is used to associate the element to any particular property defined by the user (geometrical, physical or both). This last is useful to handle the boundary conditions for the adapted mesh. Objects of the face and edge classes store similar data than elements, as well as the number of elements that share them. This information is needed when coarsening, to decide whether an entity has or not to be unrefined. If all the elements that share a face (or an edge, e.g. edge `e2` in Fig.1a) are unrefined (see elements marked with an asterisk), then the parent entity (if the entity does have a parent, in this particular case edge `e4`), is also unrefined by setting the iDs of its children to a *null* value and the child entities are "removed" from the data structure for the current mesh by inserting their iDs in the set of free iDs for that type of entity. If not, the number of elements that share the entity (see the value of `n_sh` for edge `e1`) is diminished accordingly. An edge class object also stores the iD of the vertex that appears in its middle, which is required to apply the 1-irregular vertex refinement / coarsening constraints.

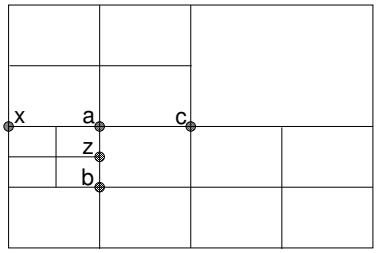


iD nodes mid. node parent n_sh

e1 ----> {a,z} -----> {null} ----> {e3} ----> 2
e2 ----> {a,y} -----> {null} ----> {e4} ----> 1
e3 ----> {a,b} -----> {z} ----> {null} ----> 2
e4 ----> {a,c} -----> {y} ----> {e5} ----> 2
e5 ----> {x,c} -----> {a} ----> {null} ----> 2

active_edges={e1,e2,e4,....}

(a) Elements to be unrefined are marked with an asterisk.



iD nodes mid. node parent n_sh

e1 ----> {a,z} -----> {null} ----> {e3} ----> 1
e2 ----> {null} ----> {null} ----> {null} ----> null
e3 ----> {a,b} -----> {z} ----> {null} ----> 2
e4 ----> {a,c} -----> {null} ----> {e5} ----> 2
e5 ----> {x,c} -----> {a} ----> {null} ----> 2

active_edges={e1,e3,e4,....}
free_iDs={e2}

(b) Edge e2 is *unrefined* because all the elements that share it are unrefined. Then, edge e3 is *activated*.

Figure 1: Unrefinement sequence.

B. Refinement schemes

The partitioning of the elements considered in this work is the same as used in Ríos Rodriguez et al. (2006); Rios Rodriguez et al. (2009); Ríos Rodriguez et al. (2011), i.e. it only considers regular 1:4 and 1:8 subdivisions for 2-D and 3-D elements, correspondingly. Therein, it is shown that refinement of tetrahedra through the shortest diagonal of the inner octaedron shows a good trade-off between the required computational effort and the geometrical quality of the resulting elements. **The algebraic quality measure η introduced by Liu and Joe (1994),**

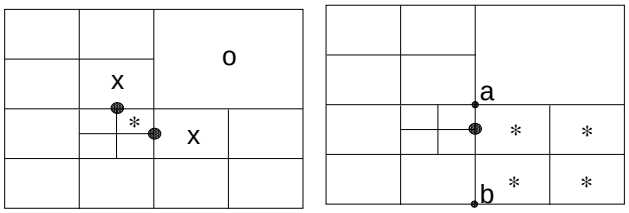
$$\eta(T) = \frac{12(3V)^{2/3}}{\sum_{i=1\ldots6} l_i^2} \tag{1}$$

was used to this end, where V is the volume of tetrahedron T and l_i is the length of its i-edge.

C. Refinement and unrefinement constraints

A smooth size change among neighbour elements in the mesh is desired because of its influence in the condition number of the stiffness matrix of the finite element method formulation (Shewchuck, 2002). **To ensure this smooth size change, we use the 1-irregular vertex refinement constraint introduced by Babuska and Rheinboldt (1978), which is also used by Remacle et al. (2002); Greaves (2003); Popinet (2003); Jasak and Gosman (2004). This constraint** states that *no more than one hanging node should be shared among neighbour elements through the common edge to which the hanging node belongs*. This means that if the element marked with an asterisk in Fig.(2.a) is to be refined, it is required that elements marked with a cross be refined first. This in turns implies that the element marked with a circle be refined formerly. As it is noted, the addition of elements to be refined because of applying this constraint induces a recursive logic for the refinement function. In 3-D the neighbourhood among elements through edges and faces as well as the refinement of *orphan* edges on triangular faces have to be considered (see Ríos Rodriguez et al. (2011) for more details). An orphan edge is one which is not obtained by the refinement of another edge.

Elements are selected to be unrefined based on the chosen criteria. Then it is necessary to guarantee that the unrefinement of these elements circumscribes to the 1-irregular vertex constraint. As can be seen in Fig.(2.b), if any of the edges of an element selected to be unrefined (marked with an asterisk) has an irregular vertex, then that element should be excluded from the list of elements to be unrefined. If not, the adapted mesh would have an edge ($\overline{ab}$ in Fig.2.b) with two irregular vertices. In conclusion, the 1-irregular vertex constraint may exclude elements that were initially marked to be unrefined while may include some elements that were not initially marked to be refined.



(a) Refinement constraint. (b) Unrefinement constraint.

Figure 2: Description of the 1-irregular vertex constraint applied at the refinement and coarsening stages.

This bias the adaptation algorithm towards refinement rather than unrefinement, which is sensible on the grounds of solution accuracy.

The coarsening algorithm considers that an element can be unrefined if only all its brothers are also marked to be unrefined. In Fig.(2.b) it is seen that because one of the brothers cannot be unrefined, none of them can.

A final constraint is imposed on the level of refinement / unrefinement for the elements. If an element belongs to the maximum level set by the user then it can not be further refined and if it belongs to the base mesh level, it can not be unrefined.

D. Adaptation algorithm

The adaptive solution of the problem follows a classical mesh adaptation algorithm. That is, given a tetrahedral mesh and proper initial and boundary conditions for the problem, the fluid flow equations are solved for a fixed number of time steps (*nsteps*). The time step size Δt is computed based on a CFL-like condition for compressible flows, based on an estimate of the maximum wave speed and element geometry. Then, the regions of the mesh that need to be refined are marked according to the following criterion, which is based on the magnitude of the Mach number gradient computed for the element. The choice of the Mach number as the variable to drive the adaptation procedure is based on the fact that it enables to capture both shock waves and contact discontinuities. Jumps in the density gradient could have also been chosen to this end. Since linear finite elements are used, the gradient for an element is constant and it is computed by derivation of the corresponding shape functions. All the elements whose gradients magnitude times theirs size are equal to or greater than a percentage of the maximum corresponding value for all the elements in the mesh are marked to be refined,

$$c_1 \leq \frac{\parallel \nabla_i M \parallel \cdot h_i}{\max_i(\parallel \nabla_i M \parallel \cdot h_i)} \tag{2}$$

where c_1 is a constant set beforehand by the user, h_i is a measure of the element size (i.e. in this work it is considered to be the length of the longest edge

for the element) and $\| \nabla_i M \|$ is the magnitude of the Mach number gradient computed for the element. The accurate choice of c_1 mostly depends on the user's experience.

The adapted mesh is generated by iteratively applying the former procedure until a maximum level of refinement prescribed by the user is attained. This limit in the number of refinement levels must be considered because there is no stopping criterion if discontinuities exists in the solution and Eq.(2) is used to select the elements to be refined (Löhner, 2008). As the base mesh is refined, the state computed by the solver is linearly interpolated and the boundary conditions are updated. **Since a linear finite element formulation is used, the linear interpolation of the variables to the new vertices of the mesh ensures conservation in a natural manner and no numerical diffusion is introduced by the adaptation scheme (Löhner, 2008).** When the maximum level of refinement is attained the interpolated state is used as the initial condition to resume flow computation. If it is not required to refine the current mesh, flow computation is resumed from the last computed state. The strategy does not consider the unrefinement of the base mesh.

After the solution is advanced *nsteps* time steps, the selection criterion given by Eq.(2) is applied to the last computed solution and elements are marked to be refined again. Also, elements are marked to be unrefined if

$$c_2 \geq \frac{\| \nabla_i M \| \cdot h_i}{\max_i(\| \nabla_i M \| \cdot h_i)} \tag{3}$$

where c_2 must be less than c_1. **The choice of the local absolute error mesh adaptation criterion given by Eqs.(2) and (3) is to achieve the equidistribution of the error in the mesh. As it is stated in Löhner (2008), this mesh optimality criterion is most often used for transient problems. Since a linear finite element approach is considered in this work, the element local error ϵ^h_{el} can be written as,**

$$\epsilon^h_{el} \propto h \| \nabla u \| \tag{4}$$

where h is a measure of the element size and u is a problem variable that enables to capture the flow features of interest. On the other hand, the choice of the values for c_1 and c_2 depends on various factors such as the mesh adaptation frequency and the flow variable used to capture the features of interest. Usually, both values must be monitored during simulation for little adjustment if needed. The c_1 value is chosen in order that refined regions are "wide" enough so that discontinuities always move inside these regions between two adaptation steps. Also, as it is stated in subsection C of this work, we have to consider that refined regions propagate due to the 1-irregular vertex constraint (i.e, over-refinement). Finally, the value for c_2 is chosen in order to ensure that unrefinement of the mesh only takes place when discontinuities are far enough, so that solution is smooth (thus avoinding conservation losses due to unrefinement (Löhner, 2008; Löhner and Baum, 1992)).

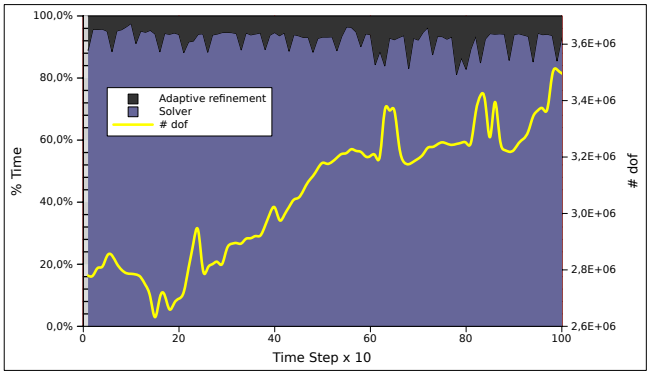


Figure 3: Relative cost of solver and adaptive refinement computations measured as clock time percentage during the simulation (number of degrees of freedom are shown on the right scale).

The adaptation strategy assumes that all those elements which satisfy Eq.(3) should be unrefined up to the base mesh level, as long as the 1-irregular vertex unrefinement constraint is already satisfied. Then, an unrefinement loop is considered. If an element can effectively be coarsened, it is replaced by its parents in the current mesh. Otherwise, it is scheduled for a potential unrefinement in the next iteration of the unrefinement loop. If it is not possible to go further with the coarsening, the loop is interrupted. After that, the refinement loop begins as described for the first adaptation step. If no elements are refined or coarsened, the flow computation resumes from the last computed state. The value for the mesh adaptation frequency *nstep* is set equal to 10 time steps of the flow solver for the problem presented in this work. This choice is based on the fact that in practice the adaptation of the mesh takes just a small fraction of the overall simulation time, i.e. approximately from 5 to 10 per cent (Ríos Rodriguez et al., 2011; Ripley et al., 2004). This result was also confirmed in this work (see Fig.3), wherein measurements of clock time during the first one thousand steps of simulation show that the fraction of time required to adapt the mesh ranges between these values.

III. Setup of the Numerical Modelization

The nozzle under consideration corresponds to the subscale VOLVO S1 TOP contour, which was designed with the geometrical contour of the Vulcain nozzle

Table 1: Subscale VOLVO S1 nozzle characteristics:

Area ratio(ε)	20
Nozzle length (L_n)	350 [mm]
Throat diameter (D_t)	67.1 [mm]
Nozzle exit diameter (D_e)	300 [mm]
Design feeding pressure (P_0)	5 [Mpa]

Table 2: Stagnation conditions in the combustion chamber used in the simulations.

P_0	ρ_0	T_0
$\{0.44; 2.1\}$ [MPa]	$\{3.8; 20.4\}$ [kg/m^3]	300 [K]

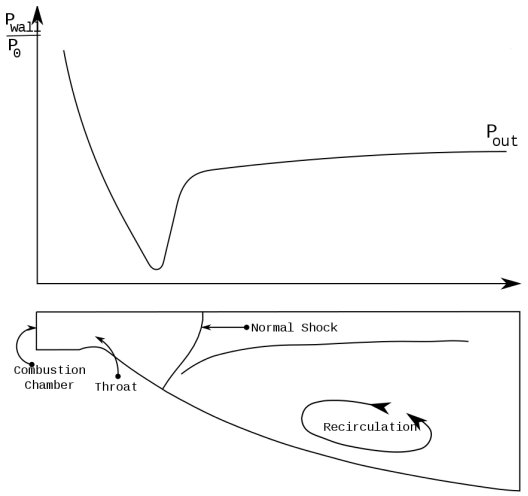
in order to study the side load phenomena (Ostlund, 2002). The characteristics of this subscale nozzle are given in Table 1.

The fluid domain is discretized with linear tetrahedral finite elements. The gasdynamics Euler equations with a consistent Streamline Upwind Petrov-Galerkin stabilization technique (Brooks and Hughes, 1982b; Franca et al., 1992; Tezduyar et al., 2006) together with the shock-capturing method (Tezduyar and Senga, 2004) are solved using a Domain Decomposition Method (DDM) with an Additive Schwarz Method (ASM) as preconditioner. **Further details about the fluid flow equations and the solution method can be consulted in the PhD thesis of Garelli (2011).**

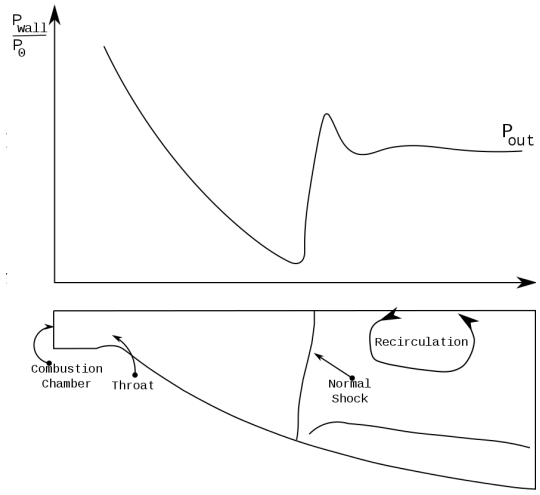
A time-varying boundary condition is imposed at the inlet of the nozzle. The stagnation pressure is linearly increased according to Table 2 from $t_0 = 0$ [sec] to $t_f = 1 \cdot 10^{-2}$ [sec] during 4000 time steps while ρ_0 is computed from the state equation since T_0 is constant. The time step size is set equal to $Dt = 2.5 \cdot 10^{-6}$ [sec] so the Courant Number (Co) ranges between 1.2 and 1.5, being the stability criterion assured for an implicit θ-method time integrator. An absorbent / dynamic boundary condition is set at the outlet with nonlinear constraints imposed via Lagrange Multipliers or a Penalty Method (Paz et al., 2010). This boundary condition allows to reduce the extension of the computational domain therefore decreasing the computational cost. Finally a velocity slip condition is applied at the wall of the nozzle.

Numerical simulations assume the nozzle is operated in over-expanded conditions because the maximum feeding pressure used in the simulations (1.85 [Mpa]) is lower than the design feeding pressure (5 [Mpa]). As a consequence, an internal shock wave develops. The shape and position of the internal shock changes as a function of the feeding pressure.

Figure (4) schematically describes the flow transition as well as the pressure ratio P_{wall}/P_0 distribution in the nozzle, P_{wall} being the pressure at the wall, P_0



(a) Separated Flow.



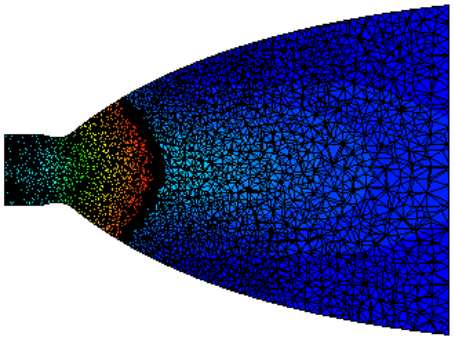
(b) Attached Flow.

Figure 4: Schematic diagram of the flow transition during startup.

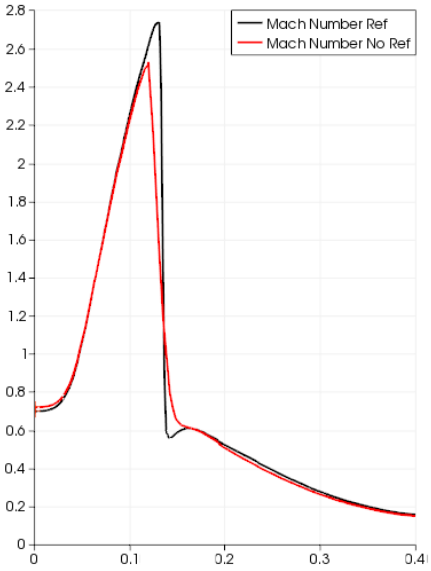
is the total pressure in the combustion chamber and P_{out} is the ambient pressure outside the nozzle. At low pressure ratios P_0/P_{out} (see Fig.4.a), the flow is detached from the wall and the outlet gases go through the center line of the nozzle, producing a recirculation zone near the wall. When the pressure ratio increases (see Fig.4.b), the flow pattern changes and the strong shock forces the outlet gases to attach to the nozzle wall. Meanwhile, the recirculation zone is located at the center line of the nozzle and the pressure there is lower than P_{out}. In this work both over-expanded conditions will be simulated and the pressure distributions on the wall and the axis will be reproduced.

IV. Numerical Results

To study the transition phenomena of the flow inside the rocket nozzle, the problem is started from a steady condition corresponding to the stagnation values $P_0 =$

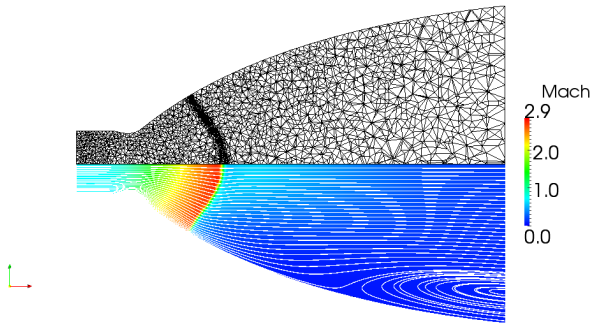


(a) Adapted mesh for the initial condition.

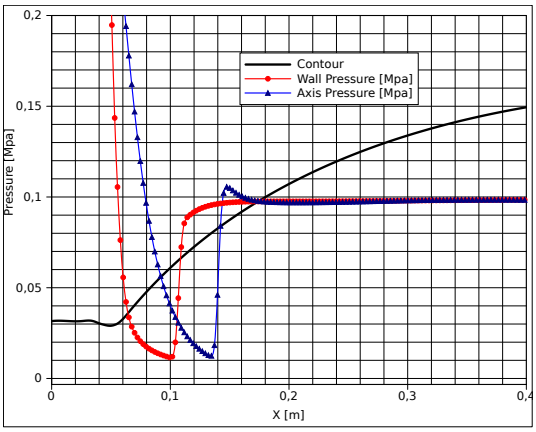


(b) Comparison of the Mach number distribution along the axis of the nozzle.

Figure 5: Improvement in the steady (initial) condition due to mesh adaptation.



(a) Streamlines and Mach number distribution.



(b) Pressure distribution on the wall and the axis of the nozzle.

Figure 6: Flowfield characterization for ($P_0 = 0.44$ [MPa]).

0.44 [MPa], $\rho_0 = 3.8$ [kg/m^3] and $T_0 = 300$ [K] in the combustion chamber, computed on the base mesh. This latter consists of 213622 tetrahedra and 38854 vertices. After the steady state is reached the mesh is refined with a Mach number-based gradient error indicator and then it is converged again to a steady condition.

Figure (5.a) shows the adapted mesh for the steady state condition and Fig.(5.b) the Mach number distribution computed along the axis of the nozzle, with and without mesh adaptation. The improvement in the sharpness of the shock due to the refinement of the mesh is clear.

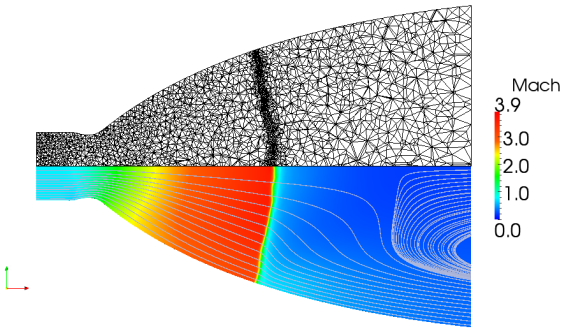
The adapted mesh shown in Fig.(5.a) has 475086 elements and 136169 vertices. The refined region can be extended or contracted by changing the values of the parameters c_1, c_2 and the numbers of refinement levels. In this problem, two levels of refinement are used and we set the values $c_1 = 0.3$ and $c_2 = 0.05$.

From this condition the pressure at the nozzle inlet is linearly increased in time according to the procedure described in the previous section (see Table 2). As a consequence, the shock wave starts to move towards the outlet and the pressure distribution on the nozzle wall and the axis modify, changing the flow pattern.

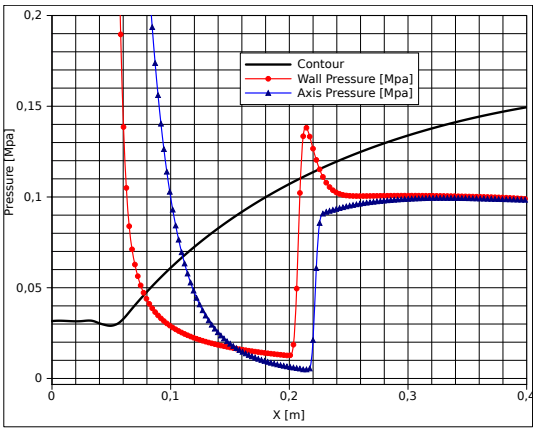
At low pressure ratios, the flow is detached from the wall after the shock and the outlet gases go through the center of the nozzle, just as explained in Fig.(4). Near the wall, the pressure rapidly reaches the ambient condition and a recirculation zone emerges, as it is shown by the streamlines in Fig.(6.a).

For a pressure value of $P_0 = 1.1$ [MPa] the flow pattern starts to change and now the pressure on the axis, right after the shock is lower than the ambient pressure and a vortex is produced, as shown in Fig.(7). At the nozzle wall there appears a pressure peak after the shock, thus producing the outlet gases to go attached to the wall. The adapted mesh for this condition consists of 584239 elements and 289828 vertices.

Finally, for $P_0 = 2.1$ [MPa] the flow pattern has fully transitioned. The pressure in the axis after the shock is lower than the ambient pressure producing a

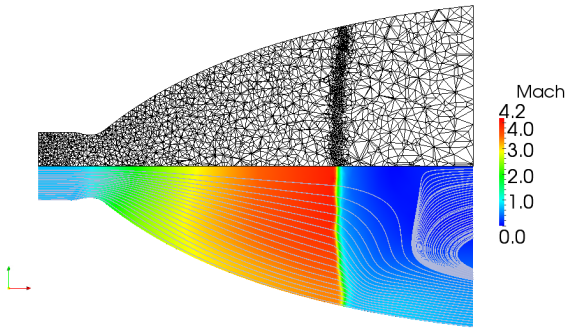


(a) Streamlines and Mach number distribution.

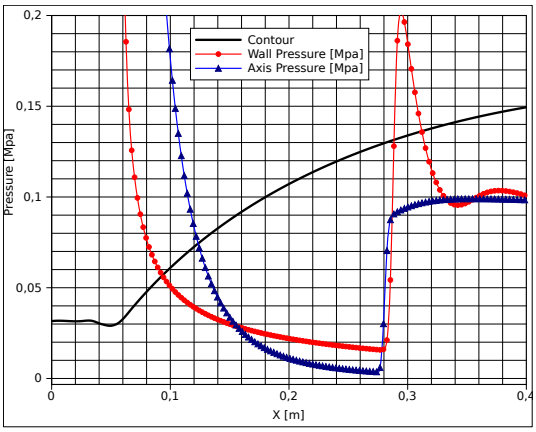


(b) Pressure distribution on the wall and the axis of the nozzle.

Figure 7: Flowfield characterization for ($P_0 = 1.1$ [MPa]).



(a) Streamlines and Mach number distribution.



(b) Pressure distribution on the wall and the axis of the nozzle.

Figure 8: Flowfield characterization for ($P_0 = 2.1$ [MPa]).

large recirculation region, as shown in Fig.(8). On the nozzle wall there is a strong pressure peak after the shock so that the outlet gases go attached to the wall. The adapted mesh has 556076 elements and 289828 vertices.

It should be noted that if a uniformly refined mesh with the same resolution as the adapted meshes had been used for the whole simulation, approximately 13 million elements would have been required, i.e. assuming two levels of homogeneous refinement of the base mesh.

Figure (9) allows to analyze the quality of the elements across the refined regions of the mesh for condition $P_0 = 2.1$ [MPa]. The quality of the elements is evaluated with the algebraic metric given by Eq.(1). The refined region at the shock wave position in the divergent of the nozzle is apparent. It can be stated that the quality of the elements in the refined region is not worse than that of the elements without refinement located in the neighbourhood of this region. As it is discussed in Rios Rodriguez et al. (2009), if the shortest diagonal strategy is used for refining in 1:8 a tetrahedron, the quality of the mesh decreases at the first refinement iteration and then keeps constant.

On the other hand, Fig.(10) shows the volume of the elements along the axis of the nozzle. It can be seen that the 1-irregular vertex constraint extended to 3-D guarantees a smooth size change across regions with different levels of refinement.

The flow transition can also be characterized by the relationship between the ratio $P_{t,i}/P_{out}$, where $P_{t,i}$ is the total pressure right ahead of the shock wave, and the nozzle pressure ratio P_0/P_{out}. In the work of Moriñigo and Salvá (2008), the internal flow of the subscale optimized J2-S nozzle is characterized analyzing the evolution of this relationship.

Figure (11) shows the evolution of the ratio $P_{t,i}/P_{out}$ for the subscale S1 nozzle analyzed in this work. Computed values are represented with dots while the continuous line is a fourth order polynomial fit. When pressure $P_{t,i}$ is higher than the outlet pressure, the main flow is detached from the wall and goes through the center of the nozzle. The onset of the flow transition occurs when $P_0/P_{out} \approx 9.2$ and $P_{t,i} \approx P_{out}$. Thereafter $P_{t,i} \leq P_{out}$ and the recirculation zone originates. A similar behaviour is shown in Moriñigo and Salvá (2008) for the start-up of the subscale optimized

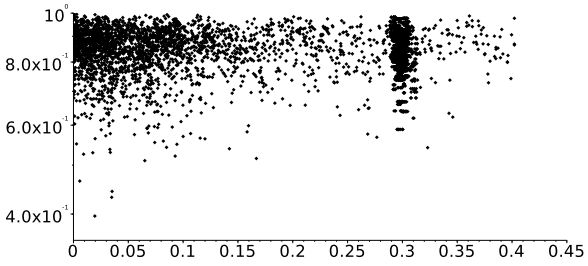


Figure 9: Mean-ratio shape measure along the axis of the nozzle for condition $P_0 = 2.1$ [MPa].

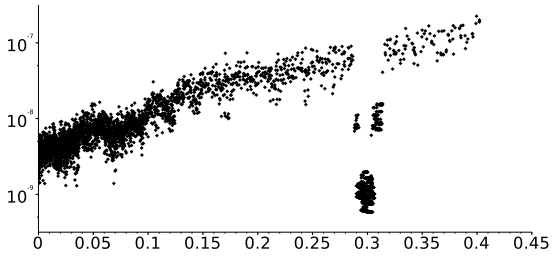


Figure 10: Elements volume along the axis of the nozzle for condition $P_0 = 2.1$ [MPa].

J2-S nozzle, where it is shown that the backflow on the axis arises as soon as $P_{t,i}$ reaches P_{out}, so the critical NPR can be determined.

V. CONCLUSIONS

As it is mentioned in the introduction, this work is a first step on the understanding of the interaction process between internal shock waves and the flow transition inside of a rocket nozzle during the start-up of the engine or when it is operated under strongly over-expanded conditions. The main objective of this work is to validate the flow solver and the unsteady mesh adaptation strategy. Numerical simulations were carried out using the PETSc-FEM software. The mesh adaptation strategy helped both to sharply resolve the shock wave pattern and to keep the computational costs as low as possible. An equivalent uniform grid as fine as the adapted ones would have demanded much higher computational resources, considering that two levels of refinement were used.

The pressure distribution along the wall and the axis were analyzed for different nozzle pressure ratios, wherewith the change in the internal flow pattern was clearly identified. Also, the streamlines were plotted in order to characterize the internal flow. This change in the flow pattern during over-expanded conditions has been reported as the origin of side loads (Ostlund, 2002).

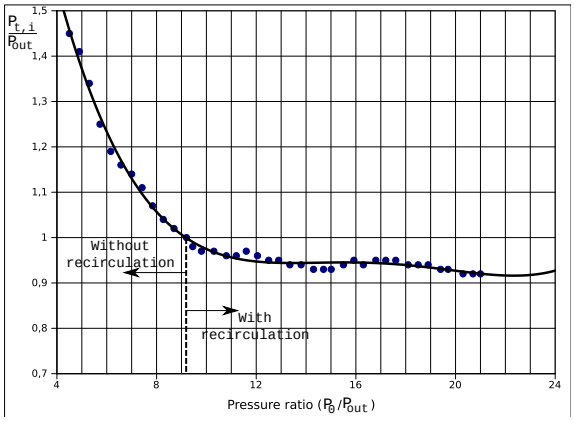


Figure 11: Total to outlet pressure ratio ($P_{t,i}/P_{out}$) at the central axis, just ahead of the shock wave, for a time varying P_0/P_{out}.

Future work aims to add a strategy developed by Garelli et al. (2010) in order to consider the structural response to the fluidynamic loads. To carry on this kind of 3D simulation, we need a very fine mesh near the wall in order to have a well resolved boundary layer and a very fine mesh where the shocks waves form, but as the shock wave moves inside of the nozzle a wide region has to be refined in order to accurately resolve the shock waves, generating a huge and unwieldy mesh. The use of transient adaptive refinament will allow to reduce the size of the problem, keeping a well resolved region near the wall and shock waves. Also, viscous effects must be included in order to analyze the interaction of the boundary layer with the shock waves, for which an hexaedral mesh may be preferably used.

Acknowledgements

This work has received financial support from Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, PIP 5271/05), Universidad Nacional del Litoral (UNL, Argentina, grants CAI+D 2009-65/334, CAI+D-2009-III-4-2), Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT-1506/2006, PICT-1141/2007, PICT-0270/2008). Authors made extensive use of *Free Software* as GNU/Linux OS, GCC/G++ compilers, Boost and STL libraries, Octave, Paraview. In addition, many ideas from these packages have been inspiring to them.

References

I. Babuska and W.C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

Boost C++ Libraries. Boost Software, `http://www.boost.org`, 1998-2012.

A.N. Brooks and T.J.R. Hughes. Streamline Upwind/Petrov Galerkin methods for advection dominated flows. In *Third Internat. Conf. of Finite Element Methods in Fluid Flow*, pages 283–292, Banff, Canada, June 1980.

A.N. Brooks and T.J.R. Hughes. Streamline Upwind/Petrov Galerkin formulations for convective dominated flows with particular emphasis on the imcompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32:199–259, 1982a.

A.N. Brooks and T.J.R. Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982b.

L.P. Franca, S.L. Frey, and T.J.R. Hughes. Stabilized finite element methods: I. application to the advective-diffusive. *Computer Methods in Applied Mechanics and Engineering*, 95:253–276, 1992.

L. Garelli. *Fluid Structure Interaction using an Arbitrary Lagrangian Eulerian formulation.* Phd. thesis, FICH, National University of Litoral, December 2011.

L. Garelli, R.R. Paz, and M.A. Storti. Fluid-structure interaction study of the start-up of a rocket engine nozzle. *Computers & Fluids*, 39:7:1208–1218, 2010.

L. Garelli, R.R. Paz, H.G. Castro, M.A. Storti, and L.D Dalcin. *Computational Fluid Dynamics: Theory, Analysis and Applications*, chapter Fluid Structure Interaction and Galilean Invariace, pages 1–40. Nova Science Publishers, 2011.

D. Greaves. A quadtree adaptive method for simulating fluid flows with moving interfaces. *J. Computational Physics*, 194 (1):35–56, 2003. ISSN 0021-9991. doi: `http://dx.doi.org/10.1016/j.jcp.2003.08.018`.

T.J.R. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics: Iii. The generalized streamline operator for multidimensional advective-diffusive systems. *Comput. Methods Appl. Mech. Engrg.*, 58(3):305–328, 1986a.

T.J.R. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics: Iv. A discontinuity-capturing operator for multidimensional advective-diffusive systems. *Comput. Methods Appl. Mech. Engrg.*, 58(3):329–336, 1986b.

H. Jasak and A. D. Gosman. Automatic resolution control for the finite-volume method, part 2: Adaptive mesh refinement and coarsening. *Numerical Heat Transfer, Part B: Fundamentals*, 38(3):257–271, 2004. doi: 10.1080/10407790050192762.

A. Liu and B. Joe. On the shape of tetrahedra from bisection. *Mathematics of Computation*, 63(207): 141–154, July 1994.

R. Löhner. *Applied CFD Techniques: An Introduction Based on Finite Element Methods.* John Wiley & Sons Inc., 2008.

R. Löhner and J.D. Baum. Adaptive h-refinement on 3D unstructured grids for transient problems. *Int. Journal for Num. Methods in Fluids*, 14:1407–1419, 1992.

J. Mattingly and Hans Von Ohain. *Elements of propulsion: gas turbines and rockets.* AIAA, 2nd edition, 2006.

J. Moriñigo and J. Salvá. Numerical study of the start-up process in an optimized rocket nozzle. *Aerospace Science and Technology*, 12:6:485–486, 2008.

G. Oates. *Aerothermodynamics of gas turbine and rocket propulsion.* AIAA, 3rd edition, 1997.

J. Ostlund. *Flow processes in rocket engine nozzle whit focus on flow separation and side-loads.* Licentiate thesis, Royal Intitute of Technology, 2002.

R R Paz, M Storti, and L Garelli. Local absorbent boundary condition for nonlinear hyperbolic problems with unknown riemann invariants. *Computers & Fluids,*, 40:52–67, 2010.

S. Popinet. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. *Journal of Computational Physics*, 190: 572–600, 2003.

S.S. Rao. *Engineering optimization.* Wiley and Sons, 1996.

J. Remacle and M.S. Shepard. Algorithm Oriented Mesh Database. *Int. J. Numer. Meth. Engng.*, 58:349–374, 2003. URL `http://dx.doi.org/10.1002/nme.774`.

J. Remacle, X. Li, N. Chevaugeon, and M.S. Shephard. Transient mesh adaptation using conforming and non conforming mesh modifications. In *Proc. 11th Int. Meshing Roundtable, Sandia National Laboratories*, September 15-18 2002.

G. A. Rios Rodriguez, N. M. Nigro, and M. A. Storti. An h-adaptive unstructured mesh refinement strategy for unsteady problems. *Latin*

American Applied Research, 39:137 – 143, 06 2009. ISSN 0327-0793.

G.A. Ríos Rodriguez, E.J. López, N.M Nigro, and M. Storti. Refinamiento adaptativo homogéneo de mallas aplicable a problemas bi- y tridimensionales. In A.E. Larrateguy, editor, *Mecánica Computacional*, volume XXIV, pages 2365–2385, Buenos Aires, Noviembre 2005. AMCA.

G.A. Ríos Rodriguez, M. Storti, and N.M Nigro. Refinamiento adaptativo en problemas no estacionarios. In A. Cardona, editor, *Mecánica Computacional*, volume XXV, pages 1163–1176, Santa Fe, Noviembre 2006. AMCA.

G.A. Ríos Rodriguez, Mario A. Storti, Ezequiel J. López, and Sofía S. Sarraf. An h-adaptive solution of the spherical blast wave problem. *International Journal of Computational Fluid Dynamics*, 25(1):31–39, January 2011. ISSN 1061-8562. doi: 10.1080/10618562.2010.543418.

R.C. Ripley, F .-S. Lien, and M.M. Yovanovich. Adaptive unstructured mesh refinement of supersonic channel flows. *International Journal of Computational Fluid Dynamics*, 18(2):189–198, 2004. doi: 10.1080/10618560310001634168.

B. Schäling. *The Boost C++ Libraries*. Xml Press, 2011.

Herbert Schildt. *C++ The complete reference*. McGraw-Hill Osborne Media, 3rd edition, 1998.

J. R. Shewchuck. What is a good linear finite element? Interpolation, conditioning, anisotropy and quality measures. `http://www.cs.cmu.edu/ jrs/jrspapers.html` (Link last retrieved 17 December 2012), 2002.

Standard Template Library. Silicon Graphics, Inc., `http://www.sgi.com/tech/stl`, 1993-2012.

E.V. Sonzogni, A.M. Yommi, N.M. Nigro, and M.A. Storti. A parallel finite element program on a Beowulf cluster. *Adv. Eng. Softw.*, 33(7–10):427–443, July / October 2002.

M.L. Staten. Selective refinement of two and three-dimensional finite element meshes. Master's thesis, Department of Civil Engineering, Brigham Young University, 1996.

M. Storti. Aquiles Cluster at CIMEC, 2005-2010. `http://www.cimec.org.ar/aquiles` (Link last retrieved 27 December 2010).

M. Storti, N. Nigro, R. Paz, L. Dalcín, L. Battaglia, E. López, and G.A. Ríos Rodriguez. *PETSc-FEM, A General Purpose, Parallel, Multi-Physics FEM Program*. CIMEC-CONICET-UNL, 1999-2010. `http://www.cimec.org.ar/petscfem`.

G. Sutton and O. Biblarz. *Rocket propulsion elements*. John Wiley and Sons, 7th edition, 2001.

T. Tezduyar and M. Senga. Determination of the shock-capturing parameters in supg formulation of compressible flows. In Tsinghua University Press & Springler-Verlag, editor, *Computational Mechanics WCCM IV, Beijing, China 2004.*, 2004.

T.E. Tezduyar and M. Senga. Stabilization and shock-capturing parameters in SUPG formulation of compressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195:1621–1632, 2006.

T.E Tezduyar, S. Sathe, R. Keedy, and K. Stein. Space-time finite element techniques for computation of fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 195:17-18:2002–2027, 2006.

M.J.L. Tuner. *Rockets and spacecraft propulsion*. Springer, 2nd edition, 2006.