JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 44, No. 1, January–February 2007

Centered and Upwind Multigrid Turbulent Flow Simulations of Launch Vehicle Configurations

Enda Dimitri Vieira Bigarella*
Instituto Tecnológico de Aeronáutica, 12228-900 São José dos Campos, SP, Brazil
João Luiz F. Azevedo†
Instituto de Aeronáutica e Espaço, 12228-903 São José dos Campos, SP, Brazil
and
Leonardo Costa Scalabrin‡
University of Michigan, Ann Arbor, Michigan 48109

DOI: 10.2514/1.23843

The paper discusses results obtained with a finite-volume code that simulates viscous, turbulent flows for 3-D, adaptive, unstructured meshes. The implementation uses a cell-centered, face-based data structure. A fully explicit, second-order accurate, five-stage, Runge–Kutta time stepping scheme is used to perform the time marching of the flow equations. Spatial discretization of the Reynolds-averaged Navier–Stokes equations can be performed with second-order centered or upwind schemes. Automatic grid refinement routines are considered to adapt the original mesh. A sensor based on density gradients selects the volumes to be refined. The code is able to handle tetrahedra, hexahedra, wedges, and pyramids. A full multigrid scheme is available to accelerate convergence to steady state. Coarse grid levels are constructed through an agglomeration procedure. One- and two-equation turbulence models are implemented to include the turbulent effects into the numerical formulation. The mentioned features integrated in one single code allow the Brazilian aerospace program to simulate complex flow conditions for sounding rockets and satellite launch vehicles with accuracy and reasonable computational resources. Good agreement with theoretical or experimental results is obtained with the present numerical tool.

Nomenclature

a	=	speed of sound
$\boldsymbol{C}$	=	convective operator
Cp	=	pressure coefficient
$\boldsymbol{D}$	=	artificial dissipation operator
d	=	minimum distance to the wall
e	=	total energy per unit volume
e_i	=	internal energy
f	=	source term of the multigrid method
k	=	specific turbulent kinetic energy
p	=	static pressure
$\boldsymbol{P}_e$	=	inviscid flux vector
$\boldsymbol{P}_v$	=	viscous flux vector
$\boldsymbol{Q}$	=	vector of conserved properties
$\boldsymbol{q}$	=	heat flux vector
RHS	=	right-hand side operator
S	=	absolute value of the mean strain-rate tensor
$\boldsymbol{S}$	=	area vector
S_{ij}	=	mean strain-rate tensor component
T	=	static temperature
u, v, w	=	Cartesian velocity components
$\boldsymbol{V}$	=	viscous operator
$\boldsymbol{v}$	=	Cartesian velocity vector
x, y, z	=	Cartesian coordinates
α	=	angle of attack
$\alpha_1 \ldots \alpha_5$	=	Runge–Kutta control parameters
γ	=	ratio of specific heats
Δt	=	time step
κ	=	von Karman constant
μ	=	dynamic viscosity coefficient
ν	=	kinematic viscosity coefficient
$\tilde{\nu}$	=	modified Spalart–Allmaras eddy-viscosity coefficient
ρ	=	density
τ	=	viscous stress tensor
Φ	=	gradient ratio for limiter computation
ψ	=	control volume limiter
ω	=	turbulent dissipation
Ω	=	absolute value of the mean rotation tensor
Ω_{ij}	=	mean rotation tensor component

Subscripts

f, k	=	face index
i, m	=	grid control volume indices
ℓ	=	laminar property
L, R	=	interface left and right properties
t	=	turbulent property
∞	=	freestream property

Superscripts

(m)	=	current grid of the multigrid method
n	=	time instant
*	=	dimensional property

Presented as Paper 5384 at the 22nd AIAA Applied Aerodynamics Conference and Exhibit, Providence, RI, 16–19 August 2004; received 12 March 2006; revision received 25 June 2006; accepted for publication 26 June 2006.

*Ph.D. Student, Centro Técnico Aeroespacial.
†Currently Director for Space Transportation and Licensing, Brazilian Space Agency. Associate Fellow AIAA.
‡Ph.D. Student, Department of Aerospace Engineering.

I. Introduction

THE paper discusses results obtained using a finite-volume method on 3-D unstructured meshes to simulate turbulent viscous flows over typical aerospace configurations. The numerical tool was developed by the CFD group at Instituto de Aeronáutica e Espaço (IAE) to aid the design of aerospace vehicles. One such aerospace configuration of interest to IAE is the first Brazilian Satellite Launch Vehicle (VLS). The VLS launcher is composed of a

central body and four strap-on boosters [1]. An illustrative sketch of the vehicle is presented in Fig. 1. This is the first time that such thorough analyses of the VLS aerodynamics at representative transonic and supersonic flight conditions are performed using a viscous, turbulent CFD tool.

The finite-volume computational code solves the compressible Reynolds-averaged Navier–Stokes (RANS) equations. A fully explicit, second-order accurate, five-stage, Runge–Kutta time-stepping scheme is used to perform the time march of the flow equations. For flux calculations on the volume faces, a Jameson centered scheme [2] plus explicitly added artificial dissipation terms [3], or a Roe flux-difference splitting scheme [4] can be used. In the latter case, second-order accuracy is achieved through a multidimensional limited [5] MUSCL-type [6] reconstruction scheme. An extension of the original multidimensional limiter formulation of [5] is proposed to keep high accuracy at generic cell types and to guarantee convergence. Furthermore, computationally faster implementation of the Roe scheme and its associated limiters is also proposed. The current methodology allows for large computational resource savings while maintaining the expected level of accuracy. The implementation uses a cell-centered, face-based data structure, and the code can use meshes with any combination of tetrahedra, hexahedra, wedges, and pyramids. Boundary conditions are set through the use of ghost cells attached to the boundary faces.

A 3-D automatic mesh adaptation technology is also available. The CFD group at IAE already has experience with 2-D mesh refinement techniques [7–9]. Some of these ideas are currently extended for the 3-D case. Basically, the refinement technique divides the original element in new ones by splitting its constitutive faces. This splitting operation, however, allows for the presence of hanging nodes, which clearly creates some additional complexity for the coding. This additional complexity, nevertheless, is considered a worthy tradeoff for increasing the overall mesh quality. The implemented routines handle meshes composed of the previously described cell element types.

Advanced eddy-viscosity turbulence models are available to include the turbulence effects into the RANS equations. Viscous simulations at high Reynolds numbers are typical for aerospace applications, such as the ones of interest to IAE. Numerical simulations of such flight conditions which do not consider turbulence effects may have limited practical application. To obtain useful viscous simulations results, the Spalart–Allmaras one-equation [10] and the SST two-equation [11] turbulence models are chosen. These models are suitable for external aerodynamics applications and they can predict flow separation with acceptable levels of accuracy. The CFD group at IAE has some previous experience with such closures for turbulent aerospace flow simulations [12–14].

A full multigrid (FMG) scheme is included to achieve better convergence rates for the simulations. To build the mesh sequence for the multigrid procedure, an agglomeration scheme based on cell or node seeds is used. The CFD group of IAE has experience with such technique [12] in other 2-D numerical codes. A robust and consistent method for 3-D turbulent flow simulations is proposed and included into the present numerical formulation. This methodology allows for successful simulations of high-Reynolds number turbulent flows for highly stretched grids at very acceptable costs.

Extensive validation of this code has already been initiated, and one is referred to [9,14,15] for a careful analysis of some validation results. The mentioned features integrated in one single code allow the Brazilian aerospace program to simulate complex flow conditions for sounding rockets and satellite launch vehicles with accuracy at reasonable computational resource expenses [9,14,15] without resorting to the high costs associated with investments in commercial flow solvers. In other words, this is a unique simulation tool developed within the Brazilian aerospace research program, which is comparable to few other commercial tool options. Furthermore, an in-house development of such a tool also adds flexibility to the assessment of the aerospace configurations of interest to IAE. To demonstrate the mentioned capabilities in an actual application context, simulation results obtained for the VLS configuration, as well as other typical aerospace test cases using the present code, are discussed in this paper. Turbulent viscous transonic and supersonic flows are considered. The numerical results obtained show good agreement with the experimental data and they represent all the relevant aerodynamic features observed in experimental tests.

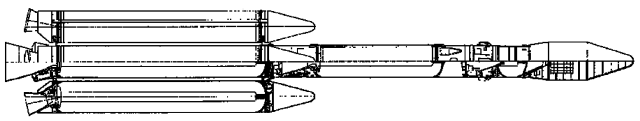


Fig. 1 VLS geometry overview.

II. Theoretical Formulation

The objective of the CFD group at IAE is to develop the capability of simulating 3-D, viscous turbulent flows over general launch vehicle configurations. An account of the theoretical formulation of the current 3-D unstructured grid flow solver is presented in the forthcoming discussions.

A. RANS Equations

The flows of interest in the present context are modeled by the 3-D compressible Reynolds-averaged Navier–Stokes (RANS) equations. These equations can be written in dimensionless form, assuming a perfect gas, as

$$\frac{\partial \boldsymbol{Q}}{\partial t} + \nabla \cdot (\boldsymbol{P}_e - \boldsymbol{P}_v) = 0 \tag{1}$$

with the following definitions

$$\boldsymbol{Q} = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{Bmatrix}, \quad \boldsymbol{P}_e = \begin{Bmatrix} \rho \boldsymbol{v} \\ \rho u \boldsymbol{v} + p\hat{\imath}_x \\ \rho v \boldsymbol{v} + p\hat{\imath}_y \\ \rho w \boldsymbol{v} + p\hat{\imath}_z \\ (e+p)\boldsymbol{v} \end{Bmatrix}, \quad \boldsymbol{P}_v = \frac{1}{Re} \begin{Bmatrix} 0 \\ \tau_{xi}\hat{\imath}_i \\ \tau_{yi}\hat{\imath}_i \\ \tau_{zi}\hat{\imath}_i \\ \beta_i \hat{\imath}_i \end{Bmatrix} \tag{2}$$

where $i = x$, y, or z are the indices used within the Einstein indexing notation; and $\hat{\imath} = \{\hat{\imath}_x, \hat{\imath}_y, \hat{\imath}_z\}$ is the Cartesian coordinate unit vector. The other relations can be given as

$$\begin{aligned} \tau_{ij} &= (\mu_\ell + \mu_t)\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\frac{\partial u_m}{\partial x_m}\delta_{ij}\right], \\ q_j &= -\gamma\left(\frac{\mu_\ell}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial (e_i)}{\partial x_j} \end{aligned} \tag{3}$$

and $\beta_i = \tau_{ij}u_j - q_i$, where u_i is the Cartesian velocity component, x_i is the Cartesian coordinate, and δ_{ij} is the Kronecker delta. In the previous definitions, μ_ℓ is the molecular dynamic viscosity coefficient, computed by the Sutherland's law [14], and μ_t is the eddy-viscosity coefficient, computed by the chosen turbulence model. The dimensionless pressure, p, can be calculated from the perfect gas equation of state as

$$p = (\gamma - 1)\left[e - \frac{1}{2}\rho(u^2 + v^2 + w^2)\right] \tag{4}$$

In this work, all properties are made dimensionless according to a set of dimensional reference variables provided by the user. The necessary dimensional reference variables are composed of a reference length D^*_{ref}, a reference speed V^*_{ref}, a reference dynamic viscosity coefficient $\mu^*_{\ell\text{ref}}$, a reference temperature T^*_{ref}, and a reference density, ρ^*_{ref}. The dimensionless properties are defined according to [16]. The user must also provide the gas properties, namely, the constant of the gas R^*, the specific heat at constant volume Cv^*, the specific heat at constant pressure Cp^*; the Prandtl number Pr, and the turbulent Prandtl number Pr_t. The gas R^*, Cp^*, and Cv^* properties are also made dimensionless with the provided

reference variables. Furthermore, the Reynolds number is defined as $Re = \rho^*_{\text{ref}} V^*_{\text{ref}} D^*_{\text{ref}} / \mu^*_{\ell\text{ref}}$. One should observe that the Euler equations are obtained from the RANS equations as the Reynolds number approaches infinity.

B. Turbulence Modeling

The present work is mainly interested in high Reynolds number simulations of flows over complex aerodynamic configurations. Two turbulence closures are chosen in the present context, namely, the Spalart–Allmaras [10] (SA) one-equation model and the Menter SST [11] two-equation model. Both closures are particularly suited for aerodynamic flow simulations and separation prediction [11]. Furthermore, they are also less restrictive in relation to the grid refinement near the wall than other two-equation models such as the k–ϵ family of models [11].

Both models are solved according to the finite-volume approach. The convective term is discretized using a simplified first-order upwind scheme, and the diffusion term is discretized using a second-order centered scheme. The time march is performed using the implicit Euler scheme. One should observe that the use of an implicit scheme for the time march, in an unstructured mesh, leads to a sparse linear system. The solution for this system of equations is obtained using the biconjugate gradient method [17]. More implementation details on the time integration of the turbulence modeling equations can be found in [9].

Finally, because a measure of the turbulent kinetic energy is used in the SST model, that turbulence quantity should be included in the viscous terms of the RANS equations. Thus, some terms are redefined as

$$\tau^{\text{new}}_{ij} = \tau_{ij} - \frac{2}{3} Re \rho k \delta_{ij}, \qquad \beta^{\text{new}}_i = \beta_i + (\mu_\ell + \sigma_k \mu_t) \frac{\partial k}{\partial x_i} \tag{5}$$

and the static pressure is redefined as $p^{\text{new}} = p - (\gamma - 1)\rho k$. The inclusion of the turbulent kinetic energy as previously shown is unusual in the CFD community. However, these operations augment the physical accuracy of turbulent viscous flow simulations because some turbulent terms extracted from the Reynolds averaging procedure are brought back to the formulation not only via the eddy-viscosity coefficient. This is specially important when higher Mach number flow conditions are considered, in which the turbulent kinetic energy can be comparable to the local mean-flow kinetic energy [18].

III. Numerical Formulation

The forthcoming subsections describe the currently adopted finite-volume method and the temporal discretization of the governing equations.

A. Finite-Volume Discretization

The finite-volume method is used to obtain the solution of the RANS equations. The formulation of the method is obtained by an integration of the flow equations in a finite volume. The application of Gauss' theorem for each finite volume yields

$$\int_{V_i} \frac{\partial \boldsymbol{Q}}{\partial t} \, \mathrm{d}V + \int_{S_i} (\boldsymbol{P}_e - \boldsymbol{P}_v) \cdot \mathrm{d}\boldsymbol{S} = 0 \tag{6}$$

where the outward-oriented area vector is defined as $\boldsymbol{S} = \{S_x, S_y, S_z\}$. The discrete value of the vector of conserved variables for the ith control volume is defined as the mean value of the conserved variables in the volume as

$$\boldsymbol{Q}_i = \frac{1}{V_i} \int_{V_i} \boldsymbol{Q} \, \mathrm{d}V_i \tag{7}$$

Hence, the final form of the finite-volume formulation for the RANS equations can be written for an elementary volume and assuming a stationary mesh as

$$\frac{\partial \boldsymbol{Q}_i}{\partial t} = -\frac{1}{V_i} \sum_{k=1}^{\text{nf}} (\boldsymbol{P}_{e_k} - \boldsymbol{P}_{v_k}) \cdot \boldsymbol{S}_k \tag{8}$$

where nf is the number of faces which form the control volume. The code is able to handle grids composed of tetrahedra, hexahedra, wedges, pyramids, or a mix of these types of elements. The previous equation also indicates that the integral is discretized assuming the fluxes to be constant on the faces.

B. Time Integration

The integration in time of Eq. (8) is performed using a five-stage Runge–Kutta-type scheme [2,19] such as

$$\boldsymbol{Q}_i^{(0)} = \boldsymbol{Q}_i^n, \qquad \boldsymbol{Q}_i^{(\ell)} = \boldsymbol{Q}_i^{(0)} - \alpha_\ell \frac{\Delta t_i}{V_i} \text{RHS}_i^{(\ell-1)}, \qquad \boldsymbol{Q}_i^{n+1} = \boldsymbol{Q}_i^{(5)} \tag{9}$$

where $\ell = 1 \cdots 5$, and the residue, RHS, is defined as

$$\text{RHS}_i = \boldsymbol{C}_i - \boldsymbol{V}_i - \boldsymbol{D}_i \tag{10}$$

Here, $\boldsymbol{C}_i$, $\boldsymbol{V}_i$ and $\boldsymbol{D}_i$ are, respectively, the convective operator, the viscous operator, and the artificial dissipation operator calculated for the ith control volume. These operators are calculated according to the spatial discretization scheme and they are detailed in the forthcoming sections. The α_ℓ coefficients are $1/4, 1/6, 3/8, 1/2$, and 1 for $\ell = 1, \cdots, 5$, respectively. The viscous operator is calculated only on the first stage of the Runge–Kutta scheme to save computational resources.

The time step for each volume, Δt_i, is calculated assuming a constant Courant–Friedrichs–Lewy (CFL) number throughout the computational domain. Hence,

$$\Delta t_i = \text{CFL} \frac{\ell_i}{|\boldsymbol{v}_i| + a_i} \tag{11}$$

where a_i is the speed of sound, $|\boldsymbol{v}_i|$ is the magnitude of the local flow velocity, and ℓ_i is the characteristic length, in the ith cell. The characteristic length is set as the smallest distance between the control volume centroid and the centroids of each face that forms the cell.

C. Spatial Discretization

Both centered and upwind schemes are available in the present numerical method for the computation of the convective fluxes.

1. *Centered Scheme*

Centered schemes require the explicit addition of artificial dissipation terms to control nonlinear instabilities that may arise in the flow simulation. The centered spatial discretization of the convective fluxes, $\boldsymbol{C}_i$, in this scheme is proposed in [2]. The convective operator is calculated as the sum of the inviscid fluxes on the faces of the ith volume as

$$\boldsymbol{C}_i = \sum_{k=1}^{\text{nf}} \boldsymbol{P}_e(\boldsymbol{Q}_k) \cdot \boldsymbol{S}_k, \qquad \boldsymbol{Q}_k = \frac{1}{2}(\boldsymbol{Q}_i + \boldsymbol{Q}_m) \tag{12}$$

where $\boldsymbol{Q}_i$ and $\boldsymbol{Q}_m$ are the conserved properties in the ith and mth cells, respectively, that share the kth face.

The artificial dissipation operator is built by a blend of undivided Laplacian and biharmonic operators. In regions of high-pressure gradients, the biharmonic operator is turned off to avoid oscillations. In smooth regions, the undivided Laplacian operator is turned off to maintain second-order accuracy. A numerical pressure sensor is responsible for this switching between the operators. The expression for the artificial dissipation operator is given by

$$\boldsymbol{D}_i = \sum_{k=1}^{\text{nb}} \left\{ \frac{1}{2}(\boldsymbol{A}_m + \boldsymbol{A}_i)[\epsilon_2(\boldsymbol{Q}_m - \boldsymbol{Q}_i) - \epsilon_4(\nabla^2\boldsymbol{Q}_m - \nabla^2\boldsymbol{Q}_i)] \right\} \tag{13}$$

where m represents the neighbor of the ith element, attached to the kth face, and nb is the total number of neighbors of the ith control volume. Furthermore, the quantities in Eq. (13) are defined as

$$\epsilon_2 = K_2 \max(\nu_i, \nu_m), \qquad \epsilon_4 = \max(0, K_4 - \epsilon_2), \qquad \nu_i = \frac{\sum_{m=1}^{\text{nb}} |p_m - p_i|}{\sum_{m=1}^{\text{nb}} [p_m + p_i]} \tag{14}$$

with the Laplacian term approximated by an undivided Laplacian computation, such as

$$\nabla^2\boldsymbol{Q}_i = \sum_{k=1}^{\text{nb}} [\boldsymbol{Q}_m - \boldsymbol{Q}_i] \tag{15}$$

In this work, K_2 and K_4 are assumed equal to $1/4$ and $3/256$, respectively. The $\boldsymbol{A}_i$ matrix coefficient in Eq. (13) is replaced by a scalar coefficient [3,20] such as

$$A_i = \sum_{k=1}^{\text{nf}} [|\boldsymbol{v}_k \cdot \boldsymbol{S}_k| + a_k|\boldsymbol{S}_k|] \tag{16}$$

This formulation is constructed in an attempt to obtain steady-state solutions which are independent of the time step [21].

In the multistage Runge–Kutta time integration previously described, the artificial dissipation operator is calculated only on the first, third, and fifth stages. For the inviscid calculations, the artificial dissipation operator is calculated in the first and second stages only. This approach guarantees the accuracy for the numerical solution while reducing computational costs per iteration [2]. Furthermore, the artificial dissipation model has also been integrated into the multigrid framework. To achieve lower computational costs for the multigrid cycles, only the first-order artificial dissipation model is used in the coarser-mesh levels. This operation is achieved by not computing the biharmonic term in Eq. (13), and by setting $\epsilon_2 \leftarrow \epsilon_2 + \epsilon_4$ in these coarser levels.

2. Upwind Scheme

The upwind discretization in the present context is performed according to the Roe flux-difference splitting scheme [4]. In this scheme, the inviscid numerical flux in the kth face can be written as

$$\boldsymbol{P}_{ek} = \frac{1}{2}(\boldsymbol{P}_{e_L} + \boldsymbol{P}_{e_R}) - \frac{1}{2}|\tilde{\boldsymbol{A}}_k|(\boldsymbol{Q}_R - \boldsymbol{Q}_L) \tag{17}$$

where $|\tilde{\boldsymbol{A}}_k|$ is the Roe matrix associated with the kth face normal direction, defined as

$$|\tilde{A}|(\boldsymbol{Q}_R - \boldsymbol{Q}_L) = \sum_{j=1}^{5} |\lambda_j|\delta_j \boldsymbol{r}_j \tag{18}$$

In this formulation, $|\lambda_j|$ represents the magnitude of the eigenvalues associated with the Euler equations, given as

$$|\boldsymbol{\Lambda}| = \text{diag}(|v_n|, |v_n|, |v_n|, |v_n + a|, |v_n - a|) \tag{19}$$

Similarly, $\boldsymbol{r}_i$ represents the associated eigenvectors, given by

$$\begin{aligned}
\boldsymbol{r}_1 &= [\, n_x \quad n_x u \quad n_x v + n_z a \quad n_x w - n_y a \quad n_x \Theta_1 + a(n_z v - n_y w)\,]^T, \\
\boldsymbol{r}_2 &= [\, n_y \quad n_y u - n_z a \quad n_y v \quad n_y w + n_x a \quad n_y \Theta_1 + a(n_x w - n_z u)\,]^T, \\
\boldsymbol{r}_3 &= [\, n_z \quad n_z u + n_y a \quad n_z v - n_x a \quad n_z w \quad n_z \Theta_1 + a(n_y u - n_x v)\,]^T, \\
\boldsymbol{r}_4 &= [\, 1 \quad u + n_x a \quad v + n_y a \quad w + n_z a \quad H + q_n a\,]^T, \\
\boldsymbol{r}_5 &= [\, 1 \quad u - n_x a \quad v - n_y a \quad w - n_z a \quad H - q_n a\,]^T
\end{aligned} \tag{20}$$

where $\Theta_1 = 0.5\boldsymbol{v} \cdot \boldsymbol{v}$. The δ_i terms represent the projections of the property jumps at the interface over the system eigenvectors, defined as the elements of

$$\boldsymbol{\Delta} = \boldsymbol{L}[\, \Delta\rho \quad \Delta(\rho u) \quad \Delta(\rho v) \quad \Delta(\rho w) \quad \Delta e\,]^{\text{T}} \tag{21}$$

where the left eigenvectors are the rows of the $\boldsymbol{L}$ matrix, which are defined as

$$\begin{aligned}
\boldsymbol{l}_1 &= \left[\, n_x + \tfrac{n_y w - n_z v}{a} - \Theta_4 n_x \quad \Theta_2 u n_x \quad \Theta_2 v n_x + \tfrac{n_z}{a} \quad \Theta_2 w n_x - \tfrac{n_y}{a} \quad -\Theta_2 n_x \,\right], \\
\boldsymbol{l}_2 &= \left[\, n_y + \tfrac{n_z u - n_x w}{a} - \Theta_4 n_y \quad \Theta_2 u n_y - \tfrac{n_z}{a} \quad \Theta_2 v n_y \quad \Theta_2 w n_y + \tfrac{n_x}{a} \quad -\Theta_2 n_y \,\right], \\
\boldsymbol{l}_3 &= \left[\, n_z + \tfrac{n_x v - n_y u}{a} - \Theta_4 n_z \quad \Theta_2 u n_z + \tfrac{n_y}{a} \quad \Theta_2 v n_z - \tfrac{n_x}{a} \quad \Theta_2 w n_z \quad -\Theta_2 n_z \,\right], \\
\boldsymbol{l}_4 &= \left[\, \Theta_3\Theta_1 - \tfrac{q_n}{2a} \quad -\Theta_3 u - \tfrac{n_x}{2a} \quad -\Theta_3 v - \tfrac{n_y}{2a} \quad -\Theta_3 w - \tfrac{n_z}{2a} \quad \Theta_3 \,\right], \\
\boldsymbol{l}_5 &= \left[\, \Theta_3\Theta_1 + \tfrac{q_n}{2a} \quad -\Theta_3 u + \tfrac{n_x}{2a} \quad -\Theta_3 v + \tfrac{n_y}{2a} \quad -\Theta_3 w + \tfrac{n_z}{2a} \quad \Theta_3 \,\right]
\end{aligned} \tag{22}$$

with $\Theta_2 = (\gamma - 1)/a^2$, $\Theta_3 = \Theta_2/2$, and $\Theta_4 = \Theta_1\Theta_2$. In the constant definitions, the kth subscript, which indicates a variable computed in the face, is eliminated to avoid overloading the equations with symbols. Properties in the volume faces are computed using the Roe average procedure, as detailed in [4].

In the classical form in which the Roe scheme is presented, such as in Eq. (17), the underlining argument is the numerical flux concept, as also found in other upwind scheme examples [22,23]. Therefore, each time the numerical flux is built, the inherent numerical dissipation is also evaluated. In an explicit Runge–Kutta-type multistage scheme, this fact means that the Roe matrix defined in Eq. (18) is computed in all stages. The present authors rather interpret the Roe scheme as the sum of a centered convective flux, defined by

$$\boldsymbol{C}_i = \sum_{k=1}^{\mathrm{nf}} \boldsymbol{P}_{e_k} \cdot \boldsymbol{S}_k, \qquad \boldsymbol{P}_{e_k} = \frac{1}{2}(\boldsymbol{P}_{e_L} + \boldsymbol{P}_{e_R}) \tag{23}$$

and an upwind-biased numerical dissipation contribution, that is given by

$$\boldsymbol{D}_i = \sum_{k=1}^{\mathrm{nf}} \frac{1}{2} |\tilde{\boldsymbol{A}}_k| (\boldsymbol{Q}_R - \boldsymbol{Q}_L) |\boldsymbol{S}_k| \tag{24}$$

Therefore, the attractive, cheaper, alternate computation of the numerical dissipation in the multistage scheme, as already used for the switched artificial dissipation schemes, is also extended to the upwind flux computation. To the authors' knowledge, this is a novel proposal and a unique feature of the present numerical tool.

To achieve second-order accuracy in space for the Roe scheme, linear distributions of properties are assumed at each cell to compute the left and right states in the face. Such states are represented by the L and R subscripts, respectively, in the previous Roe definitions. The linear reconstruction of properties is achieved through a MUSCL [6] scheme, in which the property at the interface is obtained through a limited extrapolation using the cell properties and their gradients. To perform such reconstruction at any point inside the control volume, the following expression is used for a generic element, q, of the conserved variable vector, $\boldsymbol{Q}$, in Eq. (1):

$$q(x, y, z) = q_i + \nabla q \cdot \boldsymbol{r} \tag{25}$$

where (x, y, z) is a generic point in the ith cell; q_i is the discrete value of the generic property q in the ith cell, which is attributed to the cell centroid; ∇q is the gradient of property q; and $\boldsymbol{r}$ is the distance of the cell centroid to that generic point. Gradients are computed with the aid of the gradient theorem [24], in which derivatives are converted into line integrals over the cell faces. In the present work, the control volume, V_i, in which to perform the gradient computation is chosen to be the ith cell itself [22]. The expressions for the reconstructed properties in the kth face can be written as

$$(q_L)_k = q_i + \psi_i \nabla q_i \cdot \boldsymbol{r}_{ki}, \qquad (q_R)_k = q_m + \psi_m \nabla q_m \cdot \boldsymbol{r}_{km} \tag{26}$$

where ∇q_i and ∇q_m are the gradients computed for the ith cell and its neighboring mth cell, respectively; ψ_i and ψ_m represent the limiters in these cells; and $\boldsymbol{r}_{ki}$ and $\boldsymbol{r}_{km}$ are the distance vectors from the ith and mth cell centroids, respectively, to the kth face centroid.

The first-order Roe scheme can be readily obtained by setting the limiter value to zero in Eq. (26). This operation is equivalent to writing $Q_L = Q_i$ and $Q_R = Q_m$ in the previous formulation. The integration of MUSCL-reconstructed schemes with the multigrid framework is simply accomplished by computing the second-order scheme in the finest grid level and the first-order one in the other coarser levels. This approach guarantees lower computational costs for the multigrid cycles while maintaining the adequate accuracy for the solution at the finest mesh level.

The limiter options that are available in the present context are the *minmod*, *superbee*, and van Albada limiters [25]. The respective 1-D definitions for these limiters are

$$\psi(\Phi) = \begin{cases} \min(\Phi, 1), \\ \max[\min(2\Phi, 1), \min(\Phi, 2)], \\ (\Phi^2 + \Phi)/(\Phi^2 + 1) \end{cases} \tag{27}$$

The total variation diminishing (TVD) [5,6] region is limited between the minmod and the superbee curves. In the previous equations, Φ is defined as the ratio between the gradients of adjacent control volumes at the interface. One should observe that the minmod and superbee limiters require the evaluation of maximum and minimum functions, which characterizes these limiters as nondifferentiable. The van Albada limiter, on the other hand, is continuous. This aspect is discussed further in the current section.

In a similar sense as discussed for the Roe upwind scheme, the usual way of computing limiters is to perform such calculation every time the new numerical flux should be updated. The limiter computation work, though, is a very expensive task, amounting to more than half of an iteration computational effort, in the present context. Therefore, the idea of freezing the limiter along with the dissipation operator at some stages of the multistage time-stepping scheme is attractive in terms of computational resource savings. Hence, this approach is adopted in the current code framework.

The current extension of the 1-D limiters to the multidimensional case is based on the work of Barth and Jespersen [5]. Azevedo et al. [22] also present some insights into this effort to the 2-D case. The present work, however, presents a further extension of the methodology discussed in [22]. This extension is aimed at allowing for the user the choice of any desired limiter formulation, as well as to solve some Barth and Jespersen [5] limiter drawbacks. One of such disadvantages is that it is not a continuous limiter; other aspects are discussed later.

The difficulty in implementing a TVD method in a multidimensional unstructured scheme is related to how to define the gradient ratio, Φ. A generalization of Φ for an unstructured grid is proposed in [15]. The proposed multidimensional gradient ratio for the kth face of the ith control volume is obtained as

$$\Phi_k = \begin{cases} \mathrm{num}^+/\mathrm{den}, & \text{if } \mathrm{den} > 0, \\ \mathrm{num}^-/\mathrm{den}, & \text{if } \mathrm{den} < 0, \\ 1, & \text{if } \mathrm{den} = 0 \end{cases} \tag{28}$$

where

$$\mathrm{den} = (q_i)_k - q_i, \qquad \mathrm{num}^+ = q_i^+ - q_i, \qquad \mathrm{num}^- = q_i^- - q_i \tag{29}$$

Here, the extrapolated property in the face, $(q_i)_k$, is given by

$$(q_i)_k = q_i + \nabla q_i \cdot \boldsymbol{r}_{ki} \tag{30}$$

and the $q_i^{\pm}$ variables can be mathematically defined as

$$q_i^+ = \max_f(q_i, q_f), \qquad q_i^- = \min_f(q_i, q_f) \tag{31}$$

The property in the fth face, q_f, is the arithmetic average of the properties in the neighboring cells, as in Eq. (12), resulting in $q_f = (q_i + q_m)/2$, where the mth cell shares the fth face with the ith cell. The advantage of the gradient ratio definition in Eq. (28) is that it can be directly used in any other limiter definition, such as the ones presented in Eq. (27). It can also be used to recast the original Barth and Jespersen [5] limiter formulation with a slight modification though. More details of this formulation can be found in [15].

The complete definition for the current multidimensional limiter is finally presented. The following algorithm is implemented for the computation of the limiter in the ith control volume:

1) The computation of the limiter in the ith cell is initiated by collecting the minimum, q_i^-, and the maximum, q_i^+, values for the generic q variable, in the sense of Eq. (31).

2) For each centroid of the kth face of the ith cell, the following steps are performed:

a) The property $(q_i)_k = q(x_k, y_k, z_k)$ in the kth face centroid is extrapolated, as in Eq. (30).

b) The gradient ratio in the face, Φ_k, necessary to compute the limiter, is obtained through Eq. (28).

c) A limiter value is computed at each face of the ith control volume.

3) The limiter value for the ith control volume is finally obtained as the minimum value of the limiters computed for the faces.

The nondifferentiable aspect of the minmod, superbee, and Barth and Jespersen [5] limiters poses some numerical difficulties in their use for practical numerical simulations. Their discontinuous formulation allows for limit cycles that hamper the convergence of upwind inviscid and viscous flow simulations to steady state [26]. One option to work around this problem is to freeze the limiter after some code iterations or residue drop, but this technique seems to not always work and to be highly problem dependent [26]. Such characteristics may also inhibit its application in actual production environment due to the need for user input in setting the limiter freezing operation for the simulation of interest. Another option is to use differentiable (or continuous) limiters instead of the ones which require maximum and minimum functions. Some examples can be found in [26], for instance. The limiter formulation in that work, however, seems to somehow pose a tradeoff between convergence and obtaining monotone (oscillation-free) steady-state solutions. The van Albada limiter, modified to include a smoothness augmentation threshold constant, is given by

$$\psi(\text{num}^{\pm}, \text{den}) = \frac{\text{num}^{\pm}(\text{num}^{\pm} + \text{den}) + \epsilon_{\text{LIM}}}{\text{num}^{\pm 2} + \text{den}^2 + \epsilon_{\text{LIM}}} \tag{32}$$

where num^+ or num^- are employed in the same sense specified in Eq. (28). Furthermore, ϵ_{LIM} is the constant limiter control, chosen as $\epsilon_{\text{LIM}} = 10^{-4}$ in the present work. This option seems to be appropriate for all aerospace cases considered by the present and other [27] development groups, always allowing machine-zero steady-state convergence for monotone numerical solutions.

3. *Viscous Flux Computation*

The viscous operator in the ith control volume is calculated as the sum of the viscous fluxes on the faces which constitute the volume

$$\boldsymbol{V}_i = \sum_{k=1}^{\text{nf}} \boldsymbol{P}_v(\boldsymbol{Q}_k) \cdot \boldsymbol{S}_k \tag{33}$$

In this case, both the conserved variable vector and its derivatives, on the face, are calculated as arithmetic averages between their corresponding values in the two volumes which contain the face. Derivatives of flow variables, for each control volume, are calculated in the standard finite-volume approach in which these derivatives are transformed, by the gradient theorem, into surface integrals around the control volume [24,28].

D. Boundary Conditions

"Ghost" volumes are used to enforce the boundary conditions. The boundary conditions for external flows implemented in the 3-D finite-volume code are solid wall conditions for viscous and inviscid flows, nonreflecting farfield, inlet and outlet, symmetry, and extrapolation conditions. Detailed discussions of the boundary conditions for the RANS equations can be found in [9]. Furthermore, detailed discussions on the boundary conditions for the turbulence model equations are found in [14].

IV. Multigrid Technique

A technique that may allow an excellent convergence acceleration for numerical methods is the multigrid procedure [29]. The mathematical concept in which the multigrid technique is based consists basically in eliminating the low-frequency errors of the finest grid by solving the problem in coarser grids. This is based on the knowledge that the time-integration methods available today can only rapidly eliminate high-frequency errors of a computational grid [30], and that the high frequencies associated with the coarse grids are approximately of the same order of magnitude of the low frequencies associated with the finest mesh.

A. Multigrid Methodology

The multigrid algorithm chosen for the present work is of the full approximation storage (FAS) type, which is the recommended method for nonlinear problems [30]. This method is based on exchanging both solution and residue values between different grid levels. It also relies on a good time marching procedure to be effective, such as the current Runge–Kutta time-stepping scheme.

The current multigrid method has been successfully validated within the present 3-D unstructured computational code for inviscid to turbulent viscous simulations, as shown in [31]. To improve the multigrid algorithm as well as the computational method, the simulations start at the coarsest grid level. Some iterations with the Runge–Kutta scheme are performed at this grid, and a high-order interpolation is performed to the next finer grid. Some multigrid cycles are then performed to improve the solution at this grid. This procedure is successively repeated until the finest grid is reached, with a good initial guess to the solution. Multigrid cycles are then performed on the finest mesh until convergence is reached. This technique is usually referred to as a full multigrid (FMG) scheme.

For the problem being solved, written in an operatorlike form, $L^{(0)}q^{(0)} = f^{(0)}$, the algorithm for a "V" cycle of the multigrid solver works as follows:

1) Presmoothing: execute n_1 iterations of the time marching procedure in the finest mesh level, $L^{(0)}q^{(0)} = f^{(0)}$.

2) For grid levels $g = 0$ to $M - 1$:

a) Residue computation: $r^{(g)} = f^{(g)} - L^{(g)}q^{(g)}$

b) Residue restriction: $r^{(g+1)} = R(r^{(g)})$

c) Solution restriction: $q'^{(g+1)} = R(q^{(g)})$

d) RHS computation: $f^{(g+1)} = r^{(g+1)} + L^{(g+1)}q'^{(g+1)}$

e) Presmoothing: execute n_1 iterations of the time marching procedure in level $g + 1$, $L^{(g+1)}q^{(g+1)} = f^{(g+1)}$.

3) Solve the problem for the coarsest grid level, $L^{(M)}q^{(M)} = f^{(M)}$, executing n_M iterations of the time marching procedure.

4) For grid levels $g = M$ to 1:

a) Solution correction: $q^{(g-1)} = q^{(g-1)} + P(q^{(g)} - q'^{(g)})$

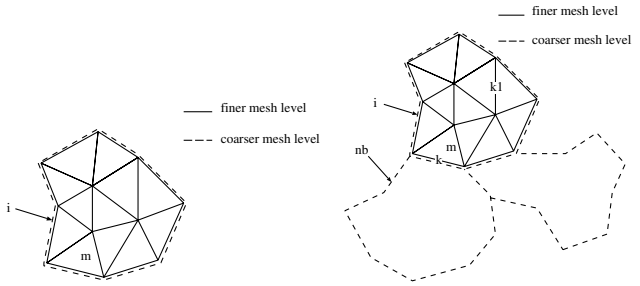
b) Postsmoothing: execute n_2 iterations of the time marching procedure in level $g - 1$, $L^{(g-1)}q^{(g-1)} = f^{(g-1)}$.

In this algorithm, r represents the RHS operator at the nth time instant, defined as the negative of the RHS operator in Eq. (10). The multigrid RHS operator, r, is also augmented by a multigrid source term, f, which represents the residue information exchange between adjacent grid levels. Moreover, q and q' represent the conserved variables at the same time instant, n, obtained from two different ways, as described in the previous algorithm. The g superscript denotes the operation for the gth grid level. The number of grid levels of this process is $M + 1$, where M is associated with the coarsest grid level and 0 with the finest one.

In the previous algorithm, exchange operators are used for the connection between two consecutive grid levels. The restriction operator exchanges information from a grid level to the next coarser mesh. The prolongation operator interpolates from a grid level to the next finer one. The operator used in the present work for the conserved property restriction is the volume weighted average. The restricted conserved properties of a coarse mesh volume are equal to the sum of the conserved properties of all the fine mesh cells that form this coarse mesh volume, weighted by their volumes. Mathematically, this operation can be written as

$$q_i^{(g+1)} = \frac{1}{V_i} \sum_{m=1}^{\text{ncv}(i)} V_m q_m^{(g)} \tag{34}$$

where the property in the ith volume of the $(g + 1)$th coarser-mesh level is a summation of the property times the volume of the $\text{ncv}(i)$ volumes of the next finer gth mesh level that belong to the ith coarser-mesh cell. This nomenclature is detailed in Fig. 2. On the other hand, the restriction of the residuals is accomplished by simple addition of the finer-mesh residuals. Thus, the residual of a coarse mesh volume



a) Restriction **b) Prolongation**

Fig. 2 Sketch of transfer operations in the multigrid scheme.

is equal to the sum of the residuals of all the finer mesh volumes that are contained in this coarse mesh volume, such as

$$r_i^{(g+1)} = \sum_{m=1}^{\mathrm{ncv}(i)} r_m^{(g)} \tag{35}$$

and the residue, r, is defined in the previous multigrid algorithm. The restriction operator for the residuals is different from the restriction operator for the conserved properties because the residuals can be interpreted as surface integrals in finite-volume schemes.

The prolongation operator is only applied to conserved property transfers. For each kth face of the mth finer-grid cell, the volume-averaged corrections corresponding to the coarser-mesh neighbors, i and nb, are multiplied by the kth face area and summed. The result is divided by the total area of the boundary surface of the mth control volume. This operation, in the $(g-1)$th mesh level, is mathematically described as

$$P(\Delta q_m^{(g)}) = \frac{1}{T_m} \sum_{k=1}^{\mathrm{nf}(m)} \left[\frac{\Delta q_i^{(g)} V_i + \Delta q_{\mathrm{nb}}^{(g)} V_{\mathrm{nb}}}{V_i + V_{\mathrm{nb}}} \right] |\boldsymbol{S}_k|, \qquad \Delta q^{(g)} = q^{(g)} - q^{\prime(g)} \tag{36}$$

The m subscript represents the mth volume of the $(g-1)$th finer grid level, $\mathrm{nf}(m)$ represents the total number of faces of this mth cell, and i and nb are the neighboring volumes of the coarser gth mesh level to the kth face. Property q' is defined in the previously described multigrid algorithm. Furthermore, V is the cell volume and T is the total area of a given cell, given by

$$T_i = \sum_{k=1}^{\mathrm{nf}(i)} |\boldsymbol{S}_k| \tag{37}$$

The nomenclature for this operation is detailed in Fig. 2. This procedure is implemented in a certain way that, if the kth face is internal to the coarser grid-level cell, as the $k1$th face in Fig. 2, then automatically $\mathrm{nb}(k1) = i(k1)$.

B. Nonflux-Term Associated Issues

During the development of the multigrid scheme in the present context, problems in the calculation of nonflux terms, such as the artificial dissipation operator, have been observed. These problems are associated with the fact that two neighboring cells may share more than one face in a given agglomerated mesh level. To simplify the explanation of such issue, the following nomenclature is used hereafter: a *face* represents a real triangular or quadrilateral planar element, whereas an *interface* represents the complete surface that separates two neighboring volumes. These two definitions are presented in Fig. 3. For the finest grid level, a face and an interface are identical to each other, because two neighboring cells are exclusively separated by a unique face. For the agglomerated grid levels, this is not true anymore and an interface between two cells may contain more than one face, as depicted in Fig. 3. In this figure, the interface between the two given cells is formed by three faces.

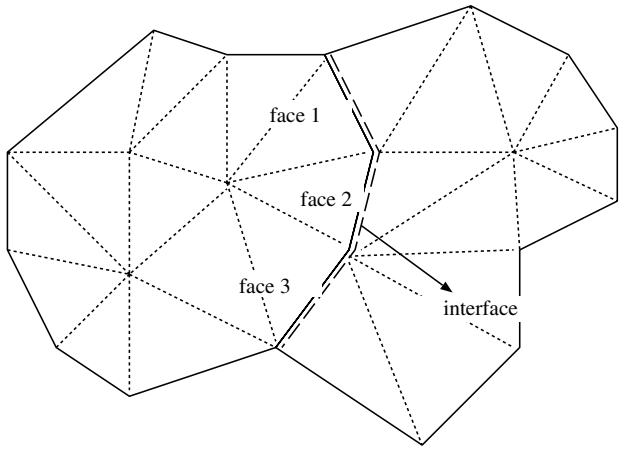


Fig. 3 Sketch of face and interface definitions.

For flux-associated terms, this fact does not represent any inconsistency because fluxes are computed through faces and they are multiplied by their surface area. This fact results in a summation of the total area of the given interface, yielding the correct flux computation for that interface. For nonflux terms, such as the artificial dissipation of the centered scheme, the computations do not take into account the face area and they must only be performed for each interface. In the present face-based code structure, one has information about faces and not about interfaces. Furthermore, storing interface information would excessively increase the memory usage of the multigrid solver. The approach in the present context to deal with this issue is to store a smaller array with the number of faces that compose a given interface between two cells. Each face receives this value, then, to be used whenever necessary. For instance, in Fig. 3, faces 1, 2, and 3 would receive the value 3 in the proper array position.

As an example, it is interesting to observe how the computation of the undivided Laplacian term, necessary for the artificial dissipation method, is performed. As already discussed, the Laplacian term is computed as

$$\nabla^2 Q_i = \sum_{k=1}^{\mathrm{nf}} [Q_k - Q_i] \tag{38}$$

where Q_i is the property in the ith volume and nf is the number of faces of the same element. What happens in the agglomerated cell computation is that, at a given interface, say the interface in Fig. 3, the $[Q_k - Q_i]$ difference is computed three times, one for each face of the interface. Nevertheless, one can clearly observe that this term is not associated with flux and, thus, this difference is added three times instead of once, which would be the correct procedure. Hence, the correct way to calculate this term would be to replace nf by nn, where nn would represent the *number of neighboring cells*, or interfaces, of the ith cell. Unfortunately, this information, along with other associated information, is not available in the code and it would be extremely costly to generate and store such data. The approach in the present context is to compute this term as indicated in Eq. (38) and divide the difference in the kth face by the number of faces that comprise the interface with the corresponding control volume. This would read, for the nm-th mesh level,

$$\nabla^2 Q_i = \sum_{k=1}^{\mathrm{nf}} \left[\frac{Q_k - Q_i}{\mathrm{ineface}(k, \mathrm{nm})} \right] \tag{39}$$

where $\mathrm{ineface}(k, \mathrm{nm})$ stores the number of similar faces for the kth face in the nm-th grid level and it is generated once in the code startup.

In the initial versions of the code which did not consider the preceding correction on the nonflux terms, strong unphysical property buildup has been observed in all simulations performed. Successful numerical results with the present multigrid method could

only be obtained with the previously described nonflux correction. This methodology is sufficient to control the unphysical property buildup generated by the unbalance of the artificial dissipation terms due to the inconsistency of the implementation of nonflux related terms for an agglomerated mesh cell.

C. Agglomeration Technique

The coarse mesh levels used by the multigrid scheme are generated by an agglomeration technique. Similar technique has been successfully implemented into a 2-D version of the present 3-D numerical formulation [12]. In the current agglomeration algorithm, a seed volume is chosen in the fine mesh and all the volumes that have at least one node in common with this seed volume are grouped to form the coarse mesh cell. Another seed volume is then selected and the agglomeration procedure continues grouping all the fine mesh volumes. It should be noted that during the agglomeration procedure, only the volumes that are not already agglomerated may be grouped to form a coarse mesh cell. This is a necessary condition to guarantee that there is no volume overlapping in the coarse mesh.

To avoid large mesh growth between successive mesh levels, there is also an option to use seed nodes instead of seed volumes for agglomeration. For meshes composed of tetrahedra, for instance, the mesh growth based on the seed-volume option may result in coarsening ratios of about 20; that is, 20 volumes of the finer mesh level are agglomerated to compose the coarser control volume. This behavior adds excessive interpolation errors for the present applications of interest. The seed-node option proved to be effective in avoiding such numerical problems, because less stringent coarsening ratios of about 8 can be obtained with its use.

Better coarse mesh quality can also be obtained if the selection of the seed volumes is not random. Therefore, a list containing all the fine mesh volumes is generated before the agglomeration procedure. For this work, the list is formed such that the first volumes are the volumes next to a solid surface, then other boundaries, and, afterward, the interior volumes. This approach is very simple to implement and adds very low additional computational cost. Although it does not necessarily provide the best agglomeration of the interior volumes, it results in good quality coarse mesh volumes close to the boundaries.

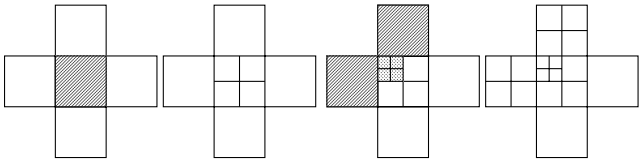
V. Adaptive Mesh Refinement

As is widely known, the quality of discrete numerical simulations is extremely dependent on the discretization mesh [32]. For high-quality numerical solutions, points concentrated in regions where the flow presents sudden variations are required. Such regions may be determined by both geometrical or aerodynamic factors. The concept of the adaptive mesh refinement is to leave to the numerical code, through flow indicatives, the responsibility of concentrating computational points by using automatic routines to modify the mesh. The adaptation can concentrate points in the identified regions by many forms. One way is to move points from flow zones that do not need them to the zones where they are needed. Another way is to create more points in these flow regions. In this work, the option of adding points is chosen because of its ease of implementation in an unstructured grid context.

The coarse mesh indicator sensor uses an undivided density gradient [7] normalized by the largest difference in density verified in the flow. This sensor can be written as

$$(\text{sensor})_i = \left[\frac{|\nabla\rho|}{\rho_{\max} - \rho_{\min}}\right]_i \tag{40}$$

The sensor automatically refers to control volumes. If it is greater than a threshold value, the volume is indicated for refinement. The types of elements handled by this code are composed of either triangular or quadrilateral faces. To allow their refinement, each face type is split into similar elements, that is, triangular faces yields child triangular faces, and the same is true with quadrilateral ones, as shown in Fig. 4a. It is interesting to point out here that the neighboring unrefined cell of a cell that is marked to be refined a second time, is marked to be refined too. This approach is currently employed to allow a smooth decrease in element size throughout the adapted mesh, as shown in Fig. 4b. The control volume is internally divided compatibly with the face division. The interested reader can find details of the division of all types of elements in [9].



a) Refinement by sensor criterion **b) Refinement by size decrease criterion**

Fig. 4 Cell refinement by different criteria. Highlighted elements are marked for refinement.

It is interesting to observe the presence of hanging nodes in the split element, as depicted in Fig. 4a. The treatment of hanging nodes is an important aspect, but with a simple implementation. The data structure of the code is face-based; therefore, it is irrelevant for the code if the node is hanging or not. The important aspect is that control volumes in this approach are no longer treated just as tetrahedra, hexahedra, wedges, or pyramids. In fact, each control volume may have the number of faces ranging from 4 (tetrahedron) to 24 (a hexahedra with all its faces divided). Moreover, one is only limited to 24 faces in the present case due to the size decrease criterion. Furthermore, because the data structure is face-based and each face stores the elements that contain it, this means that no memory is wasted. All loops are face-based and this approach makes the flux and dissipation term calculations independent of the element type.

VI. Verification Results

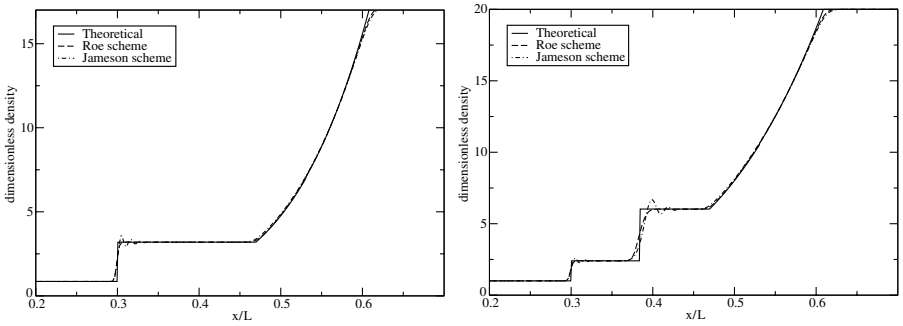
Aerodynamic flows over various aerospace configurations are simulated with the present computational tool. Results are compared to available theoretical or experimental data to assess the quality of the results that can be obtained with the numerical tool. A 1-D shock-tube problem is used to evaluate flux computation scheme results. A flat-plate flow is considered to address the turbulent flow simulation capability. The multigrid scheme is used in the simulations to accelerate convergence to steady state.

A. 1-D Shock Tube

Computations of 1-D shock-tube inviscid flow cases are considered. Numerical results are compared to the analytical solution for this problem. For the numerical simulations, an equivalent 3-D grid composed of a line of 500 hexahedra is used. The initial dimensionless density condition for the left half part of the shock tube is $\rho^L_{\text{ini}} = 1$, whereas on the right half, $\rho^R_{\text{ini}} = 20$. The reference conditions are taken in the initial state of the low-pressure side of the shock tube. Equal temperatures are assumed at both sides of the shock tube. Several other simulations with different density ratios have also been performed and the results are essentially similar to the ones presented in the forthcoming analyses. A constant dimensionless time step of $\Delta t = 10^{-5}$ is used for this transient solution, and the forthcoming plots are taken at dimensionless time $t = 0.1$.

The proposed Roe flux scheme implementation, which uses the concept of centered convective flux plus upwind artificial dissipation terms computed in alternate stages of the Runge–Kutta time marching procedure, is used. The van Albada limiter is chosen for the reconstruction process within the currently proposed multidimensional limiter implementation. The limiter computation is also performed at alternate stages of the Runge–Kutta time step. Numerical results for this formulation are compared to the ones obtained with the centered scheme.

Pressure and density distributions for the upwind and centered schemes are presented in Fig. 5. It can be observed that numerical solutions compare very well with the analytical one. No oscillations



a) Pressure **b) Density**

Fig. 5 Property distributions along the shock tube obtained for different flux schemes.

near discontinuities can be found in the upwind numerical results, whereas oscillatory behavior is found for the centered scheme solution. Further detailed analyses of the numerical scheme and limiter formulations can be found in [15]. In that work, other numerical schemes and test cases are carefully studied, including thorough mesh refinement and topology dependency study of the current numerical scheme. Successful results are obtained for shocked and boundary layer flows with various combinations of numerical schemes and mesh refinement levels. It is also interesting to remark here that the proposed alternate computation of the Roe fluxes and limiters within the Runge–Kutta scheme guarantees exactly the same solution as if computing them at all stages. However, such procedure yields a numerical scheme which is 60% faster [15].

B. Flat-Plate Turbulent Flow

A zero-pressure-gradient flat-plate flow at $Re = 7.62 \times 10^6$ and low freestream Mach number $M_\infty = 0.2$ is considered. This is an important test case because a theoretical solution is provided for the turbulent boundary layer that builds up over the flat-plate surface [33], known as the log-law solution. Simulations with both SA and SST turbulence models are included to provide a comparison of the numerical results for both closures. The hexahedra mesh about the flat plate is clustered near the flat-plate surface to guarantee the condition of $y+ \approx 1$ close to the solid boundary, which is required for integration of turbulence models to the wall. The mesh is also clustered near the flat-plate leading edge to account for the larger velocity gradients that are expected in that region. The resulting grid is composed of 60 cells within the boundary layer, and 80 cells streamwise the flat-plate length. Simulations are carried out with a standard multigrid setting of three grid levels, "V" cycles, and one Runge–Kutta smoother pass at each grid level.

Figure 6 shows the numerical boundary layers obtained for both turbulence models compared to the theoretical solution in [33] and to the experimental data in [34]. The definition of the dimensionless $u+$ and $y+$ variables in this figure can be found in [33]. One can clearly observe in Fig. 6 a striking coherence with the theoretical curve for both turbulence models. This is an indication of the correctness of the implementation of the turbulence models in the present code.

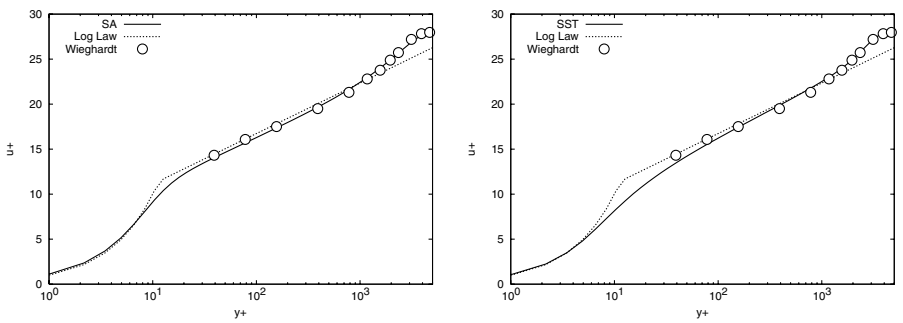
VII. VLS Results

Inviscid flows over the VLS in its first-stage (takeoff) and second-stage flight configurations are simulated for various Mach numbers using the adaptive mesh refinement routines. Turbulent viscous transonic and supersonic flows about the VLS second-stage flight configuration at various angles of attack are also considered. Because of reasonable numerical results and lower computational costs, the SA model is chosen for the forthcoming turbulent flow simulations. These flight conditions are chosen considering the dominant supersonic characteristic of the vehicle flight, and the numerical difficulties that arise at transonic simulations. Experimental data obtained through extensive high-Reynolds-number transonic and supersonic wind tunnel tests are available for this configuration [1]. Detailed description on the test setup and the experimental results for the VLS configuration can be found in [1]. Numerical results are compared to them such that the code effectiveness in the solution of realistic aerospace configuration flows can be assessed.

A. Inviscid Supersonic Flow over the First-Stage Flight Configuration

An inviscid flow at $M_\infty = 2.0$ and $\alpha = 0$ deg over the VLS is addressed. A view of the rocket nose and booster nose cap regions can be observed in Fig. 8. The mesh has approximately 106,000 nodes, and 581,000 tetrahedral volumes. In the geometry, the engine nozzles are not considered as this component is not adequately simulated with an inviscid formulation [35]. Furthermore, the currently available computational resources do not allow turbulent viscous flow simulations for the entire configuration. Despite the large number of control volumes, the mesh is not refined enough near the body, as can be observed in Fig. 8.

Figure 7 shows density contour results for the current flow case. Detached shock waves in front of the rocket nose and booster regions can be observed in this figure. The flow in the vehicle forebody presents a detached shock wave and a stagnation region in the rocket



a) SA **b) SST**

Fig. 6 Flat-plate turbulent boundary layer numerical, theoretical, and experimental results.

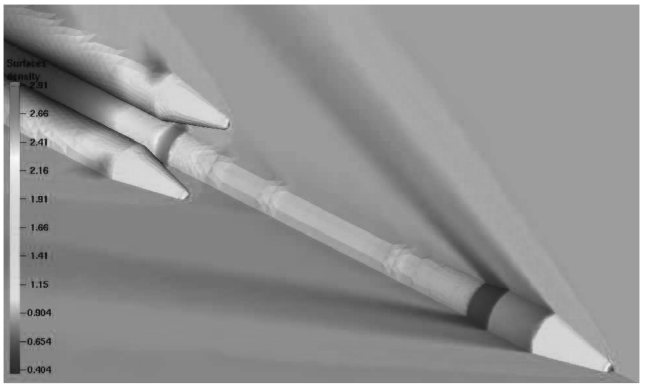


Fig. 7 Density contours over the VLS for inviscid flow simulations at $M_\infty = 2.0$ and zero angle of attack.

nose region. An expansion occurring along the conical part and over the corners of the payload fairing, and a shock wave at the end of the boattail, can also be observed. Another interesting feature in the flowfield is the high-pressure zone between the boosters, created by shock wave reflection mechanisms. The low-pressure region downstream this high-pressure zone is originated by the outward flow deflection caused by the presence of the high-pressure zone. It should be pointed out that the discussions here are presented in terms of pressure, as pressure and density are directly related by the state equation for ideal gases.

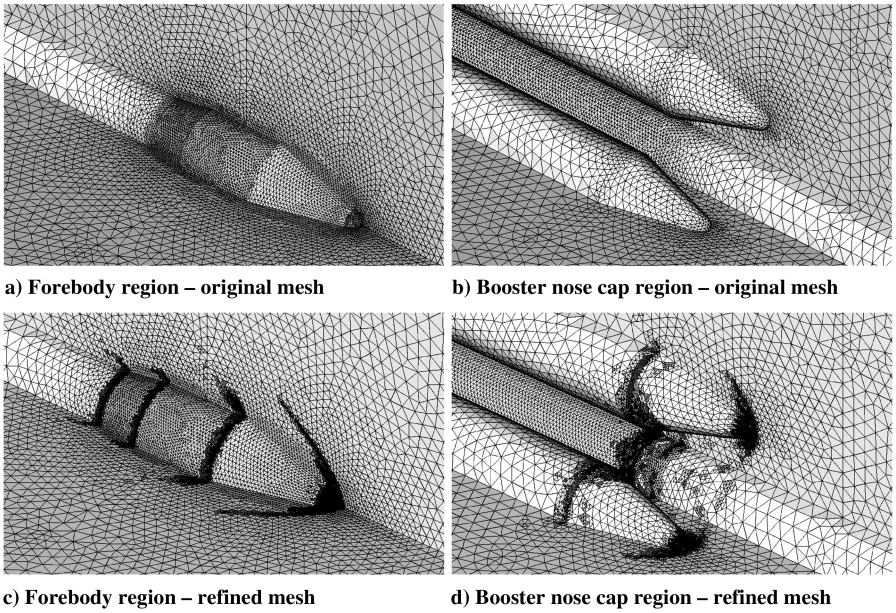
The computational mesh used in the previous results is less than adequately refined for such flight condition. The results show the need for mesh refinement in the booster nose cap and vehicle forebody regions, to adequately solve the complex shock wave structures in those regions. The adaptive mesh refinement technique is applied to the previous flow case to enhance the solution quality. Because of RAM memory restrictions, the authors limit the adaptation to only two refinement passes, although the routines are able to perform as many refinement passes as required. For the current case, the refinement threshold is set to 0.003. The first mesh refinement occurs at 5000 iterations in the original mesh, and the second one at 10,000 iterations in the refined mesh. The final mesh, after two refinement passes, has 362,740 nodes and 1,618,558 volumes, that is, about three times larger than the original coarser one. Detailed views of the adapted mesh over the vehicle forebody and booster nose cap regions are presented in Fig. 8. Clearly, the adaptation routines detect the presence of shock waves and expansions, and the mesh is consecutively refined mainly over these regions. The authors observe that this is a fully tetrahedral mesh.

Comparisons of pressure coefficient distributions on the VLS pitching plane for the nonadapted and adapted meshes, are presented in Fig. 9. It can be observed in this figure that the adaptive refinement technique increases the solution quality. This can be seen as thinner shock waves and expansions on the vehicle forebody corners. However, two mesh refinement passes are still not enough to significantly enhance the solution near the booster nose cap zone. This part of the flow is characterized by the intense interactions between the detached shock waves from the boosters. The value of the pressure coefficient in this region is overpredicted by all numerical solutions and the authors believe that only a turbulent viscous simulation could obtain better results for that region.

B. Inviscid Transonic Flow over the First-Stage Flight Configuration

Density contours over the VLS at $M_\infty = 0.9$ and $\alpha = 0$ deg are presented in Fig. 10. Interesting features of the flow are the shock waves over the payload fairing and boosters. Besides the payload-fairing shock wave, expansions in the corners of the payload fairing and a weak shock wave over the boattail can also be observed. One can also see that the flow rapidly accelerates between the boosters up to the formation of a shock wave. It should be noted that, for this transonic case, there is no high-pressure region downstream the shock wave as clearly marked as in the supersonic case. In that condition, the high-pressure region is formed by intense shock wave reflections in the region between the boosters. A previous work in the group with this test case using the same mesh, presented several problems related to solution oscillations in the domain [36]. A more refined mesh was necessary to yield results with similar quality to the present ones, which indicates the robustness of the current finite-volume code.

The adaptive mesh refinement routines are applied to the current flow case. Only one mesh refinement pass is performed, and the mesh refinement sensor threshold is set to 0.0007. The same grid used in the previous VLS first-stage supersonic simulations, as presented in Fig. 8, is also currently employed. As already discussed, this mesh is composed of 106,000 nodes and 581,000 tetrahedral cells. The adapted mesh after one refinement pass has 242,801 nodes and 768,177 elements. The mesh refinement is applied at 30,000 iterations in the original mesh. Detailed views of the adapted mesh over the vehicle forebody and booster nose cap regions are presented in Fig. 11. The mesh refinement routines detect the relevant regions of the flow, namely, stagnation in the central body and booster nose regions, the transonic shock wave over the payload-fairing



a) Forebody region – original mesh **b) Booster nose cap region – original mesh**

c) Forebody region – refined mesh **d) Booster nose cap region – refined mesh**

Fig. 8 Original and adapted meshes for inviscid flow simulations at $M_\infty = 2.0$ and zero angle of attack.

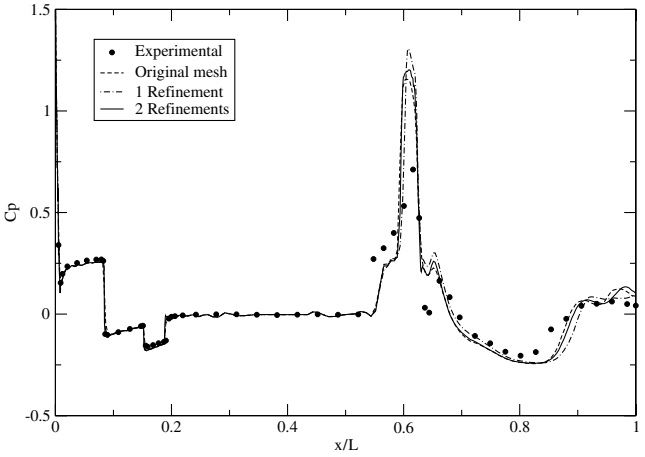


Fig. 9 Pressure coefficient distributions for inviscid flow simulations at $M_\infty = 2.0$ and zero angle of attack.

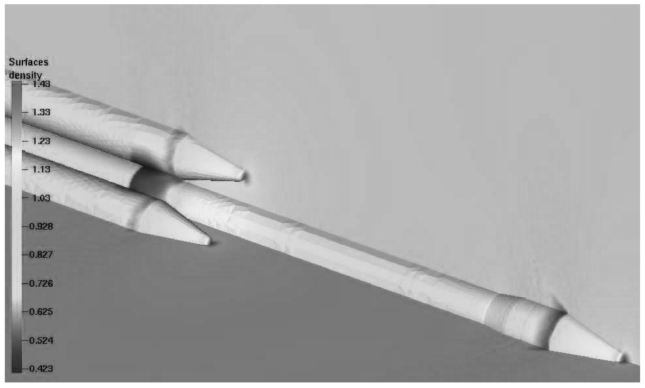


Fig. 10 Pressure contours over the VLS for inviscid flow simulations at $M_\infty = 0.9$ and zero angle of attack.

component, and the expansion zone between the boosters. Mesh density is subsequently increased in the selected regions.

Comparisons of pressure coefficient distributions in the vehicle pitching plane for the nonadapted and adapted meshes is presented in Fig. 12. The improvement in the solution quality in this case is not as good as in the supersonic case. There are only minor improvements, such as a slightly better position of the shock wave in the payload-fairing cylinder. The inviscid solution presents higher pressure peaks than the experimental data. This behavior is already expected because large boundary layer interaction, mainly for the payload-fairing shock wave, is expected. This difference is diminished using turbulence models, as depicted in the next sections.

C. Turbulent Transonic Flow over the Second-Stage Flight Configuration

Turbulent viscous flows over the VLS at $M_\infty = 0.9$, $Re = 25 \times 10^6$ and zero angle of attack are considered. The mesh used in this

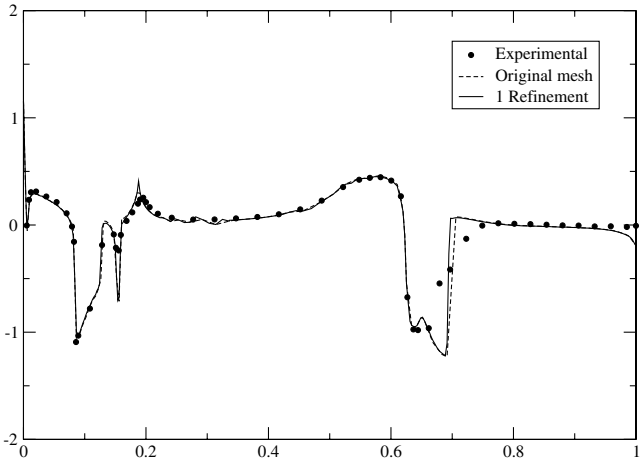


Fig. 12 Pressure coefficient distributions for inviscid flow simulations at $M_\infty = 0.9$ and zero angle of attack.

case has 100,815 nodes and 89,280 hexahedra. The hexahedra are clustered near solid walls to guarantee $y+ \approx 1$ for the interior volumes attached to the solid surface. Furthermore, about 40 hexahedra are placed within the boundary layer, to guarantee sufficient grid resolution in that region.

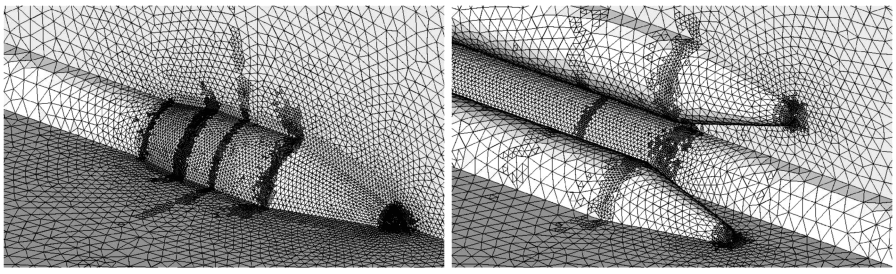
Figure 13 presents Mach number contours over the vehicle forebody. This figure evidences the presence of the boundary layer over the vehicle. A stagnation point occurs in front of the vehicle. There is a supersonic expansion over the end of the conical forebody. This causes the formation of a supersonic region on the payload cylinder which is ended by a shock wave. Because of the boundary layer, this shock wave does not reach the body. The region over the end of the boattail presents very small velocities, but the boundary layer does not separate because of the turbulent characteristics of the flow. In fact, laminar simulations of this flow condition indicate boundary layer separation.

A comparison between pressure coefficient distributions over the VLS surface for inviscid and turbulent numerical solutions, as well as experimental data, is also shown in Fig. 13. The simulation with turbulent effects present more adequate results if compared to the Euler simulation, as already expected. A more consistent solution is obtained, which better captures the strong shock wave over the payload fairing and the weak one slightly downstream of the boattail expansion corner.

D. Turbulent Supersonic Flow over the Second-Stage Flight Configuration

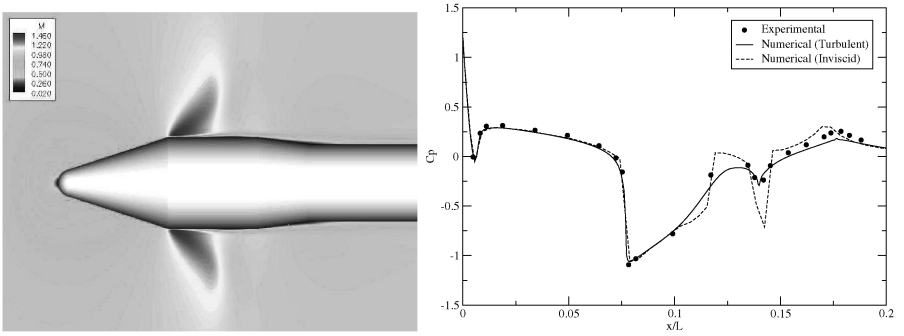
A turbulent flow at $M_\infty = 2.0$, $Re = 30 \times 10^6$ and zero angle of attack over the VLS second-stage flight configuration is simulated. The mesh used in this case has 201,565 nodes and 188,480 hexahedra. The hexahedra are clustered near the solid surface to guarantee $y+ \approx 1$, and approximately 40 cells are placed within the boundary layer, for consistent resolution of the turbulent effects in that region.

Figure 14 presents Mach number contours over the vehicle forebody. This flow case is characterized by a detached shock wave



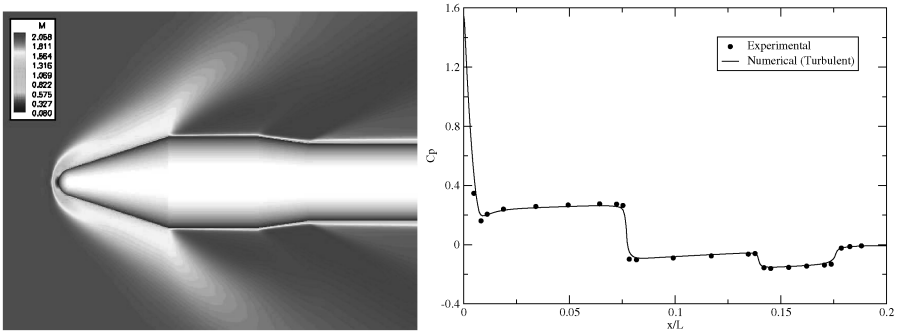
a) Forebody region **b) Forebody region**

Fig. 11 Adapted mesh near the vehicle forebody for inviscid flow simulations at $M_\infty = 0.9$ and zero angle of attack.



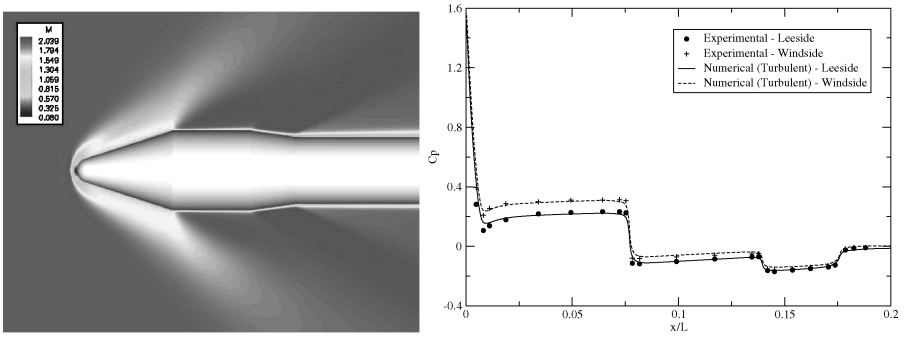
a) Mach number contours b) *Cp* distributions

Fig. 13 Numerical and experimental [1] results for the VLS at $M_\infty = 0.9$, $Re = 25 \times 10^6$ and zero angle of attack.



a) Mach number contours b) *Cp* distributions

Fig. 14 Numerical and experimental [1] results for the VLS at $M_\infty = 2.0$, $Re = 30 \times 10^6$ and zero angle of attack.



a) Mach number contours b) *Cp* distributions

Fig. 15 Numerical and experimental [1] results for the VLS at $M_\infty = 2.0$, $Re = 30 \times 10^6$ and $\alpha = 2.0$ deg.

in the vehicle nose, an expansion on the first and second corners of the payload fairing, and a shock wave at the end of the boattail. A comparison between pressure coefficient distributions over the VLS surface for the turbulent numerical solution and the experimental data is also shown in Fig. 14. There are no relevant differences between the solutions for this case, as expected.

One of the objectives of the CFD group at IAE is to determine the stability derivatives of launch vehicles for many flow conditions. This is usually performed by the determination of the pressure coefficient on the vehicle wall for different angles of attack at the flow conditions of interest. Therefore, a simulation of the VLS flying at angle of attack different from zero is a relevant condition to be tested. As launch vehicles fly at very small angles of attack, the flow over the VLS second-stage flight configuration at $M_\infty = 2.0$, $\alpha = 2.0$ deg, and $Re = 30 \times 10^6$, is considered. The same computational mesh for the zero angle-of-attack case is used in this computation. Figure 15 presents Mach number contours over the VLS forebody in the pitching plane. The detached shock wave is no longer symmetrical. The shock wave on the windside is stronger than on the leeside because of the larger deflection of the flow in that region. Furthermore, the boundary layer on the leeside is thicker than on the windside. A comparison between numerical and experimental pressure coefficient distributions over the vehicle forebody in the pitch plane is also presented in Fig. 15. As one can observe in this figure, the numerical solution once again presents good agreement with the experimental data.

VIII. Conclusions

The paper presents results obtained with a unique, 3-D, finite-volume code developed within the Brazilian aerospace program to solve the RANS equations over typical aerospace configurations. The code uses a Runge–Kutta-type scheme to perform the time march of the flow equations. The code is designed to use unstructured meshes composed of any combination of tetrahedra, hexahedra, wedges, and pyramids. The agglomeration multigrid scheme

provides large convergence acceleration for the numerical simulations. In a general manner, numerical solutions of complicated flows such as transonic turbulent flows about a typical aerospace configurations can be obtained in half the time used by the single-grid simulation.

The fluxes on the volume faces are computed by either a centered scheme plus explicitly added artificial dissipation to control nonlinear instabilities, or a second-order flux-difference-splitting upwind scheme. Application of the upwind scheme in simulations of typical aerospace configurations shows that consistent results can be obtained with this formulation. The upwind scheme presents better results in solving laminar boundary layers due to its more physically coherent artificial dissipation formulation [15]. Therefore, it guarantees further accuracy for the numerical tool with which successful aerospace application results can already be obtained.

Turbulence effects are added to the formulation by eddy-viscosity-based one- and two-equation turbulence models. The Spalart–Allmaras one-equation and the SST two-equation turbulence closures are chosen to include the turbulence effects into the RANS equations. Comparison of numerical boundary layers over a zero-pressure gradient flat-plate flow with the corresponding theoretical log-law solution shows the level of accuracy that can be obtained with the present formulation. Furthermore, the code is also able to correctly solve for more complex flows, such as transonic or supersonic turbulent flows about typical aerospace configurations. Again, good approximation between experimental and numerical results are obtained for such cases.

The integration of all discussed features within a single computational code allow robust and accurate numerical solutions for complex turbulent transonic to supersonic flows with reasonable computational resource usage. This capability is demonstrated with simulations of various typical aerospace configurations. Therefore, the main contributions of the present work are a detailed description of an accurate and robust numerical tool for high-Reynolds-number aerospace applications, a novel formulation and implementation of a MUSCL-type upwind scheme, and a thorough analysis of the VLS launch vehicle aerodynamics. The quality of the numerical solutions that can be obtained with the present computational code are demonstrated. Successful results are presented with good comparison to experimental or theoretical data. VLS flow simulations provide consistent numerical results of acceptable quality for typical transonic and supersonic flight conditions. Simulation results show good approximation to wind tunnel data. The results presented here are a good indication of the capability of simulating turbulent flows about realistic aerospace configurations that has been developed by the CFD group at IAE. These results also show the maturity of the code to be inserted in the design phase of aerospace vehicles as a reliable and accurate numerical tool.

Acknowledgments

The authors acknowledge Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, which partially supported the present work under the Integrated Project Research Grant No. 501200/2003-7. The authors are also indebted to Centro Nacional de Supercomputação at Universidade Federal do Rio Grande do Sul, which has provided some of the computational resources used for the present simulations.

References

[1] Moraes, P., Jr., and Augusto Neto, A., "Aerodynamic Experimental Investigation of the Brazilian Satellite Launch Vehicle (VLS)," *Proceedings of the 3rd Brazilian Thermal Sciences Meeting—ENCIT 90*, Vol. 1, ABCM, Itapema, SC, Brazil, 1990, pp. 211–215.

[2] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259, June 1981.

[3] Mavriplis, D. J., "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes," *AIAA Journal*, Vol. 28, No. 2, Feb. 1990, pp. 213–221.

[4] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Dierence Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, Oct. 1981, pp. 357–372.

[5] Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes," AIAA Paper 89-0366, Jan. 1989.

[6] van Leer, B., "Towards the Ultimate Conservative Dierence Scheme. 5. A Second-Order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, No. 1, July 1979, pp. 101–136.

[7] Azevedo, J. L. F., and Korzenowski, H., "Comparison of Unstructured Grid Finite Volume Methods for Cold Gas Hypersonic Flow Simulations," AIAA Paper 98-2629 June 1998.

[8] Korzenowski, H., Figueira da Silva, L. F., and Azevedo, J. L. F., "Unstructured Adaptive Grid Flow Simulations of Inert and Reactive Gas Mixtures," *Proceedings of the 14th Brazilian Congress of Mechanical Engineering*, ABCM, Bauru, SP, Dec. 1997.

[9] Scalabrin, L. C., "Numerical Simulation of Three-Dimensional Flows over Aerospace Configurations,' M.S. Thesis, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil, July 2002.

[10] Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, Jan. 1992.

[11] Menter, F. R., and Grotjans, H., "Application of Advanced Turbulence Models to Complex Industrial Flows," *Second International Conference on Advances in Fluid Mechanics*, Udine, Italy, May 1998.

[12] Strauss, D., and Azevedo, J. L. F., "Unstructured Multigrid Simulations of Turbulent Launch Vehicle Flows Including a Propulsive Jet," *Journal of Spacecraft and Rockets*, Vol. 41, No. 5, Sept.–Oct. 2004, pp. 745–753.

[13] Bigarella, E. D. V., "Three-Dimensional Turbulent Flow Simulations over Aerospace Configurations," M.S. Thesis, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil, Dec. 2002.

[14] Bigarella, E. D. V., Basso, E., and Azevedo, J. L. F., "Centered and Upwind Multigrid Turbulent Flow Simulations with Applications to Launch Vehicles," AIAA Paper 2004-5384, Aug. 2004.

[15] Bigarella, E. D. V., and Azevedo, J. L. F., "A Study of Convective Flux Computation Schemes for Aerodynamic Flows," AIAA Paper 2005-0633, Jan. 2005.

[16] Pulliam, T. H., and Steger, J. L., "Implicit Finite-Difference Simulations of Three-Dimensional Compressible Flow," *AIAA Journal*, Vol. 18, No. 2, Feb. 1980, pp. 159–167.

[17] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in C: The Art of Scientic Computing*, Cambridge Univ. Press, Cambridge, England, 1992, pp. 83–89, Chap. 2.

[18] Wilcox, D. C., *Turbulence Modeling for CFD*, 2nd ed., DCW Industries, La Cañada, CA, 1998.

[19] Jameson, A., and Mavriplis, D., "Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh," *AIAA Journal*, Vol. 24, No. 4, 1986, pp. 611–618.

[20] Mavriplis, D. J., "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes," *AIAA Journal*, Vol. 26, No. 7, July 1988, pp. 824–831.

[21] Azevedo, J. L. F., "On the Development of Unstructured Grid Finite Volume Solvers for High Speed Flows," Instituto de Aeronautica e Espaco, Report NT-075-ASE-N/92, São José dos Campos, SP, Brazil, Dec. 1992.

[22] Azevedo, J. L. F., Figueira da Silva, L. F., and Strauss, D., "Order of Accuracy Study of Unstructured Grid Finite Volume Upwind Scheme," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, May 2005 (submitted for publication).

[23] Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods," *Journal of Computational Physics*, Vol. 40, No. 2, April 1981, pp. 263–293.

[24] Swanson, R. C., and Radespiel, R., "Cell Centered and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations," *AIAA Journal*, Vol. 29, No. 5, May 1991, pp. 697–703.

[25] Hirsch, C., *Numerical Computation of Internal and External Flows. 2. Computational Methods for Inviscid and Viscous Flows*, Wiley, Chichester, 1991, Chap. 21.

[26] Venkatakrishnan, V., "Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters," *Journal of Computational Physics*, Vol. 118, No. 1, April 1995, pp. 120–130.

[27] Oliveira, G. L., "Analyse Numérique de l'Effect du Défilement des Sillages liés aux Interactions Rotor-Stator en Turbomachines, Ph.D. thesis (in French), Ecole Centrale de Lyon, Laboratoire de Mécanique des Fluides et d'Acoustique—UMR 5509, Lyon, France, March 1999.

[28] Azevedo, J. L. F., Strauss, D., and Figueira da Silva, L. F., "An Order of Accuracy Analysis for Flux-Vector Splitting Schemes on Unstructured

Grids," *Proceedings of the 15th Brazilian Congress of Mechanical Engineering—COBEM 99*, ABCM, Águas de Lindóia, SP, Brazil, Nov. 1999.

[29] Wesseling, P., "Introduction to Multigrid Methods," ICASE Report No. 95-11, NASA Langley Research Center, Hampton, VA, Feb. 1995.

[30] Fletcher, C. A. J., *Computational Techniques for Fluid Dynamics. 2. Specific Techniques for Different Flow Categories*, Springer–Verlag, Berlin, 1988, pp. 203–209, Chap. 6.

[31] Bigarella, E. D. V., Basso, E., and Azevedo, J. L. F., "Multigrid Adaptive-Mesh Turbulent Simulations of Launch Vehicle Flows," AIAA Paper 2003-4076, June 2003.

[32] Lain, K. R., Brodersen, O., Rakowitz, M., Vassberg, J. C., Tinoco, E. N., Wahls, R. A., Morrison, J. H., and Godard, J., "Summary of Data from the Second AIAA CFD Drag PredictionWorkshop (invited)," AIAA Paper 2004-0555, Jan. 2004.

[33] Tennekes, H., and Lumley, J. L., *A First Course in Turbulence*, The MIT Press, Cambridge, MA, 1972.

[34] Coles, D. E., and Hirst, E. A., "Computation of Turbulent Boundary Layers," *Proceedings of the 1968 AFOSR-IFP-Stanford Conference*, Vol. 2, Stanford Univ. Press, Palo Alto, CA, 1969.

[35] Strauss, D., and Azevedo, J. L. F., "A Numerical Study of Turbulent Afterbody Flows Including a Propulsive Jet," AIAA Paper 99-3190, June–July 1999, pp. 654–664.

[36] Scalabrin, L. C., Azevedo, J. L. F., Teixeira, P. R. F., and Awruch, A. M., "Three Dimensional Flow Simulations with the Finite Element Technique over a Multi-Stage Rocket," AIAA Paper 2002-0408, Jan. 2002.

R. Cummings
Associate Editor