1 Introduction

The streamline-upwind/Petrov–Galerkin (SUPG) formulation of compressible flows, supplemented with shock-capturing, has been successfully used over a quarter of a century. In this paper, for inviscid compressible flows, the $YZ\beta$ shock-capturing parameter, which was developed recently and is based on conservation variables only, is compared with an earlier parameter derived based on the entropy variables. Our studies include comparing, in the context of these two versions of the SUPG formulation, computational efficiency of the element- and edge-based data structures in iterative computation of compressible flows. Tests include 1D, 2D, and 3D examples. [DOI: 10.1115/1.3062968]

Keywords: inviscid compressible flow, SUPG formulation, stabilization parameter, $YZ\beta$ shock-capturing, edge-based data structure
2 Governing Equations

The 3D equations of inviscid compressible flows can be written as

\[
\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0 \quad \text{in} \quad \Omega \times [0, T_{\text{max}}]
\]

where \( \Omega \subset \mathbb{R}^3 \) and \( t \in [0, T_{\text{max}}] \). The vector of conservation variables \( \mathbf{U} \) and the vector of inviscid fluxes \( \mathbf{F} \) are given as

\[
\mathbf{U} = \rho \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ e \end{pmatrix}, \quad \mathbf{F} = \rho \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ e \end{pmatrix} + p \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
\]

Here \( \rho \) is the density, \( \mathbf{u} = [u_1, u_2, u_3]^T \) is the velocity vector, \( e \) is the total energy density, \( p \) is the pressure, and \( \delta_i \) is the Kronecker delta. The equation of state used here corresponds to the ideal gas assumption. Alternatively, Eq. (1) can be written as

\[
\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}_i \frac{\partial \mathbf{U}}{\partial x_i} = 0 \quad \text{in} \quad \Omega \times [0, T_{\text{max}}], \quad \mathbf{A}_i = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}
\]

Appropriate sets of boundary and initial conditions are assumed to accompany Eq. (3).

3 Numerical Formulation

3.1 Finite Element Formulation. We assume that we have constructed some suitably defined finite-dimensional trial solution and test function spaces \( \mathcal{S}^h \) and \( \mathcal{V}^h \). Based on that, the SUPG formulation [4,5,7] can be written as follows: find \( \mathbf{U}^h \in \mathcal{S}^h \) such that \( \mathcal{V}^h \)

\[
\int_{\Omega} \mathbf{W}^h \cdot \left( \frac{\partial \mathbf{U}^h}{\partial t} + \mathbf{A}_i \frac{\partial \mathbf{U}^h}{\partial x_i} \right) d\Omega + \sum_{e=1}^{n_e} \int_{\Gamma_e} \mathbf{N}^h \cdot \left( \frac{\partial \mathbf{W}^h}{\partial x_i} \right) d\Gamma_e + \sum_{e=1}^{n_e} \int_{\Gamma_e} \mathbf{N}^h \cdot \frac{\partial \mathbf{U}^h}{\partial x_i} d\Gamma_e = 0
\]

At each time step, the coupled nonlinear equations involving the solution variables are solved with the predictor-multicorrection algorithm described for compressible flows in Ref. [5]. An iterative technique with nodal-block-diagonal preconditioner and GMRES update method [24] is employed for solving the linear system equations. The SUPG stabilization and shock-capturing parameters are denoted by \( \tau \) and \( \tau_{\text{shoc}} \). They will be discussed in Secs. 3.2–3.4.

3.2 Stabilization Parameters. In our test computations, we compare the performance of the new stabilization parameters to 3–5, continued with Ref. [9], reached maturity in Ref. [7], and involved further adjustments in subsequent publications. The specific form of \( \tau_{\text{mod}} \) given here is from Ref. [25]:

\[
\tau_{\text{mod}} = \max(0, \tau + (\tau_0 - \tau_0))
\]

\[
\tau_{\text{mod}} = \frac{2}{1 + 2\alpha c h}, \quad \tau_0 = \frac{h}{2u_c}, \quad \tau_{\text{shoc}} = \frac{\delta_i}{(u_c)^2}
\]

where \( h \) is an element length defined as the cubic root of the element volume, and \( c, C, \) and \( u_c \) are defined as follows:

\[
\gamma = \frac{2\alpha c h}{1 + 2\alpha c h}, \quad C = \frac{u_c}{c + u_c}, \quad \frac{\partial U^h}{\partial U^h}
\]

Here \( c \) is the acoustic speed, \( \alpha \) is a parameter controlling the stability and accuracy of the time-marching algorithm (set as \( \alpha = 0.5 \) here), and \( \Delta t \) is the time-step size.

New ways of calculating the stabilization parameters are introduced in the context of the (SUPG)2 presented in Refs. [15–17] and tested in Refs. [18–20]. Here we provide from Refs. [15–17] the version we use in our computations. For this purpose, we first 125 define the unit vector \( j = \nabla \rho / \| \nabla \rho \| \). The stabilization parameter \( \tau_{\text{UGN}} \) is defined from its components corresponding to the advection- and transient-dominated limits, \( \tau_{\text{UGN}} \) and \( \tau_{\text{UGN}} \). In computing \( \tau_{\text{UGN}} \), each component of the test vector-function \( W \), the stabilization parameters \( \tau_{\text{UGN}} \) and \( \tau_{\text{UGN}} \) are defined as follows:

\[
\tau_{\text{UGN}} = \tau_{\text{UGN}} = \tau_{\text{UGN}} = \frac{\Delta t}{2}
\]

The parameters \( \tau_{\text{UGN}}, \tau_{\text{UGN}}, \) and \( \tau_{\text{UGN}} \) are calculated from their components by using the \( r \)-switch [10]:

\[
\tau_{\text{UGN}} = \left( \frac{1}{\tau_{\text{UGN}}} + \frac{1}{\tau_{\text{UGN}}} \right)^{-1/r}
\]

Typically \( r=2 \). Thus, the resulting diagonal stabilization parameter matrix \( \tau_{\text{UGN}} \) is written as

\[
\tau_{\text{UGN}} = \begin{pmatrix} \tau_{\text{UGN}} & 0 \\ 0 & \tau_{\text{UGN}} \end{pmatrix}
\]
variables. The YZβ shock-capturing was introduced in Refs. [15–17] and tested in Refs. [18–20]. Here we provide from Refs. [15–17] the version we use in our computations:

\[ v_{shoc} = \|Y^{-1}Z\| \left( \sum_{a=1}^{ned} \|Y^{-1} \frac{\partial U^h}{\partial x_j}\|^2 \right)^{\beta/2-1} \]

\[ \|Y^{-1}U\|^{-\beta} \left( \frac{h_{shoc}}{2} \right)^{\beta} \]  

where \( Y \) is a diagonal scaling matrix constructed from the reference values of the components of \( U \):

\[ Y = \begin{bmatrix} (U_1)_{ref} \\ (U_2)_{ref} \\ (U_3)_{ref} \\ (U_4)_{ref} \\ (U_5)_{ref} \end{bmatrix} \]  

\[ Z = \frac{\partial U^n}{\partial t} + A^n \frac{\partial U^n}{\partial x_j} \]  

\[ h_{shoc} = 2 \left( \sum_{a=1}^{ned} [j \cdot \nabla N_a] \right)^{-1} \]  

The parameter \( \beta \) is set as \( \beta = 1 \) for smoother shocks and \( \beta = 2 \) for sharper shocks. The compromise between the \( \beta = 1 \) and \( \beta = 2 \) selections was defined in Refs. [15–17] as the following averaged expression for \( v_{shoc} \):

\[ v_{shoc} = \frac{1}{2} (v_{shoc}^{\beta = 1} + v_{shoc}^{\beta = 2}) \]

3.4 Edge-Based Solver. Following the algebraic approach in Ref. [14], the element matrices can be disassembled into their edge contributions. For all elements sharing a given edge, one can add their terms and construct the edge matrix, which is a \( 10 \times 10 \) nonsymmetric matrix. The edge matrix is smaller than the element matrix, which is \( 20 \times 20 \), but the number of edges is always greater than the number of elements. The number of terms stored in the edge-based data structure requires less memory and fewer computations than the element-based data structure. Note, however, that different from element-based data structures, the edge-based data structures require the scatter and add of element contributions to the six edges. This is clearly an overhead, present also when you use other data structures, as for instance, sparse formats.

Further gains can be achieved when using a block-diagonal preconditioner, as in our case. There is no need to store the edge matrix block-diagonals but only the inverse of the global block-diagonal \( B_d \). This way of performing the matrix-vector products, either element or edge based, as presented in Ref. [26] for the scalar transport equation, can be used for the problem at hand here in the following manner:

\[ \mathbf{A}x = \mathbf{B}_d \mathbf{x} + \sum_{s=1}^{ned} \left( \mathbf{A}_s \right) \mathbf{x}^s \]  

\[ \mathbf{B}^T \mathbf{x} = \mathbf{B}_d^T \mathbf{B} \mathbf{x} + \mathbf{B}^T \mathbf{x} + \sum_{s=1}^{ned} \left( \mathbf{A}_s \right) \mathbf{x}^s \]  

\[ \mathbf{B}^T \mathbf{A} \mathbf{x} = \mathbf{x} + \mathbf{B}^T \sum_{s=1}^{ned} \left( \mathbf{A}_s \right) \mathbf{x}^s \]

where “ned” is the number of edges, and \( \mathbf{A}_s \) is the matrix of off-diagonal terms associated with edge \( s \).

We note that the resulting nonsymmetric edge matrix has 100 terms to be stored, minus the block-diagonal terms, which are 50.

Thus, there is only need to store half of the terms, since the inverted block-diagonals are required for preconditioning purposes.

We also note that this gain is more significant for the edge-based data structure than it is for the element-based one; because for the element-based approach only 100 of the 400 terms are in the block-diagonal and do not need to be stored.

Table 1 shows, for computation of matrix-vector products with five degrees of freedom and element- and edge-based data structures for tetrahedral meshes with \( N \) nodes, the storage requirements for the effective mass-matrix coefficients (storage), the costs of indirect addressing (i.a.), and the floating point operations (flops). According to the estimates given in Ref. [27] for tetrahedral meshes, nel=5.5N and ned=7N. Table 1 demonstrates the superiority of the edge-based data structure over the element-based one, both in memory requirements and operation count. In many cases, the edge-based data structure does not present a good balance between floating point and i.a. operations. To improve this ratio, several alternatives to the basic edge-based approach were proposed in Ref. [28], which are based on combining the already gathered data as much as possible. This idea, combined with node renumbering strategies, introduces further enhancements, as shown in Ref. [28]. The basic approach is used here because it is simpler and provides better computational performance for the problems computed.

4 Numerical Examples

All coefficients of the effective mass matrix and residual vector were calculated with the aid of a symbolic mathematical software, resulting in a code with one single loop over the elements to compute and assemble the edge-matrices and residual. The effective mass-matrix routine has around 6400 lines of code. All were calculated with the aid of a symbolic mathematical software, and ned=7N. Table 1 demonstrates the superiority of the edge-based data structure over the element-based one, both in memory requirements and operation count. In many cases, the edge-based data structure does not present a good balance between floating point and i.a. operations. To improve this ratio, several alternatives to the basic edge-based approach were proposed in Ref. [28], which are based on combining the already gathered data as much as possible. This idea, combined with node renumbering strategies, introduces further enhancements, as shown in Ref. [28]. The basic approach is used here because it is simpler and provides better computational performance for the problems computed.

4.1 1D Shock Tube. A shock tube problem is an essentially 1D flow discontinuity problem that provides a good test for compressible flows simulations. The domain is a cylindrical or rectangular tube, with a middle membrane barrier separating two initial gas states at different pressures and densities. The pressure and density can be high in one of the halves and low in the other. The well-known Sod shock tube benchmark problem is considered here. The solution contains simultaneously a shock wave, a contact discontinuity, and an expansion fan. A reference solution was obtained using a fine mesh with 1000 \times 1 \text{ cells}, each divided into five tetrahedra. This solution is in agreement with the analytical solution (see Ref. [29]). In our test computation, we use a mesh with 100 \times 1 \text{ cells}, which is shown in Fig. 1. The initial condition consists of \( p=1.0, \rho_1=0.0, \) and \( p=1.0 \) on the
4.2 2D Flow in a Channel With a Step. The problem of a wind tunnel containing a step was first described in Ref. [30]. Although it has no analytical solution, this problem is useful in testing the performance of a method in handling unsteady shock interactions in multiple dimensions. The 2D rectangular domain is three units wide and one unit high. The step is between \( x = 0.6 \) and \( x = 3.0 \), with a height of 0.2 units. As boundary condition at the step, the normal component of the velocity is set to zero. The normal component of the velocity is zero also along the upper and lower channel walls. The supersonic inflow conditions at the left boundary are \( \rho = 1.4, u_1 = 3.0, \) and \( n_2 = 0.0 \), corresponding to a Mach number of 3. Because the condition at the outflow boundary on the right is supersonic throughout the calculation, no boundary condition is specified there. The initial conditions are set equal to the inflow conditions, with \( u_1 = 0.0 \) along the left edge of the step. The mesh, which is shown in Fig. 3, has 58,509 nodes, 173,130 elements, and 290,142 edges.

The time-step size is \( 10^{-3} \), and the test duration is 2.0. For the preconditioned GMRES solver, 30 vectors are used in the Krylov basis, with a maximum of 100 cycles and a tolerance of \( 10^{-5} \). Nonlinear tolerance is \( 10^{-2} \). As reference values in Eq. (16), we use the initial condition values for the left domain. In the option denoted by \( \nu_{\text{shoc left}} \), we use the initial condition values for the right domain.

Figures 6 and 7 show the distribution of \( \delta_{b1} \) and \( \nu_{\text{shoc}} \). We note that the two parameters have the same order of magnitude (0 < \( \delta_{b1} < 2.75 \times 10^{-2} \) and 0 < \( \nu_{\text{shoc}} < 2.0 \times 10^{-2} \)) but have different distributions, with \( \nu_{\text{shoc}} \) mimicking the shock interactions. We note that they show good agreement with the benchmark solution [30].

Figures 4 and 5 show the density obtained with \( \delta_{b1} \) and \( \nu_{\text{shoc}} \). We note that they show good agreement with the benchmark solution [30].

4.3 3D Flow Around a Sphere. The sphere has a unit radius and the Mach number is 3. We consider only half of the sphere to obtain the steady-state solution. Figure 8 shows the dimensions of the problem domain.

The far-field conditions are \( \rho_\infty = 1, u_\infty = (3, 0, 0)^T, \) and \( \gamma = 6.3 \). We impose these far-field conditions at the left and upper boundaries. The flow is supersonic at the outflow boundary on the right.
and therefore no boundary condition is specified there. Along the symmetry plane and on the cylinder surface the normal component of the velocity is zero. The initial conditions are set to the far-field conditions. The mesh, shown in Fig. 9, has 15,032 nodes, 78,915 elements, and 97,809 edges.

The time-step size is $10^{-2}$, and the time-marching continues until $t=6$. For the preconditioned GMRES solver, 25 vectors are used in the Krylov basis, with a maximum of 50 cycles and a tolerance of $10^{-6}$. The number of nonlinear iterations per time step is 3. As reference values in Eq. (16), we use the initial conditions.

Figures 10 and 11 show the density obtained with $\delta_1$ and $r_{\text{shoc}}$. The two are in good agreement.

Table 2 shows, for the element- and edge-based data structures, the relative CPU times for the matrix evaluation, matrix-vector products in the GMRES iterations, and the total run times. Matrix evaluation is somewhat slower for the edge-based data structure, but this is very much compensated by the large reduction in the CPU time involved in the matrix-vector products, resulting in roughly 40% reduction in the total run time.

5 Concluding Remarks

We provided a comprehensive assessment of the stabilization and shock-capturing parameters introduced recently for SUPG formulation of compressible flows based on conservation variables. We focused on performance evaluation of the $YZ\beta$ shock-capturing parameter. We used well-known 1D, 2D, and 3D test problems with tetrahedral meshes. At each time step, the coupled
nonlinear equations involved were solved with the predictor-
multicorrector algorithm. An iterative technique with nodal-block-
diagonal preconditioner and GMRES update method was employed
for solving the linear equation system involved. We used an edge-
based data structure to store the Jacobian and perform the matrix-
vector products. In our test computations, we compared the \( YZ \theta \)
shock-capturing parameter to \( \delta' \), which was derived from its
counterpart in entropy variables. We also tested different options
for choosing the reference values used in \( YZ \theta \) shock-capturing. In
addition to being simpler and requiring less floating point opera-
tions, \( YZ \theta \) shock-capturing yields better shock quality. We pro-

---

Fig. 6 2D flow in a channel with a step. Distribution of \( \delta' \). (a) \( t=0.3 \), (b) \( t=0.6 \), and (c) \( t=1.2 \).

Fig. 7 2D flow in a channel with a step. Distribution of \( \rho_{shoc} \). (a) \( t=0.3 \), (b) \( t=0.6 \), and (c) \( t=1.2 \).

Fig. 8 3D flow around a sphere. Dimensions of the problem domain. The cylinder is located at \( x_i=8 \).

Fig. 9 3D flow around a sphere. Mesh.

Fig. 10 3D flow around a sphere. Density obtained with \( \delta' \).
vided an assessment of the computational efficiency of the edge-based structure compared with the element-based one. Although computing and storing the edge-matrices is a bit slower, matrix-vector products are computed around five times faster, reducing the total run time by about 40%.

Acknowledgment

We are indebted to Jose J. Camata, a graduate student at COPPE/UFRJ, for his help in the last stages of this work.

References


#1 Au: Please define “V-SGS” if possible.
#2 Au: Please confirm change from “the following sections” to “Secs. 3.2–3.4” here.
#3 Au: Please supply last page no. in Ref. 30.