Strong scaling of numerical solver for supersonic jet flow configurations

Carlos Junqueira-Junior[1] · João Luiz F. Azevedo[2] · Jairo Panetta[3] · William R. Wolf[4] · Sami Yamouni[5]

Abstract
Acoustics loads are rocket design constraints which push researches and engineers to invest efforts in the aeroacoustics phenomena which is present on launch vehicles. Therefore, an in-house computational fluid dynamics tool is developed in order to reproduce high-fidelity results of supersonic jet flows for aeroacoustic analogy applications. The solver is written using the large eddy simulation formulation that is discretized using a finite-difference approach and an explicit time integration. Numerical simulations of supersonic jet flows are very expensive and demand efficient high-performance computing. Therefore, non-blocking message passage interface protocols and parallel input/output features are implemented into the code in order to perform simulations which demand up to one billion degrees of freedom. The present work evaluates the parallel efficiency of the solver when running on a supercomputer with a maximum theoretical peak of 127.4 TFLOPS. Speedup curves are generated using nine different workloads. Moreover, the validation results of a realistic flow condition
AQ1 are also presented in the current work.

Keywords Computational fluid dynamics · Large eddy simulation · Strong scalability · Supersonic jet flow

1 Introduction

One of the main design issues related to launch vehicles lies on noise emission originated by the complex interaction between the high-temperature/high-velocity exhaustion gases and the atmospheric air. These emissions, which have high noise levels, can damage the launching structure or even be reflected upon the vehicle structure itself and the equipment onboard at the top of the vehicles. The resulting pressure fluctuations can damage the solid structure of different parts of the launcher or of the onboard scientific equipment by vibrational acoustic stress. Therefore, it is strongly recommended to consider the load resulted from acoustic sources over large launching vehicles during take-off and also during the transonic flight.

The authors are interested in studying unsteady property fields of compressible jet flow configurations in order to eventually understand the acoustic phenomena, which are important design constraints for rocket applications. Experimental techniques used to evaluate such flow configuration are complex and require considerably expensive apparatus. Therefore, the authors have developed a numerical tool, JAZzY [17], based on the large eddy simulation (LES) formulation [11] in order to perform time-dependent simulations of compressible jet flows. JAZzY is a compressible LES parallel code for the calculation of supersonic jet flow configurations. The large eddy simulation approach has been successfully used by the scientific community and can provide high-fidelity numerical data for aeroacoustic applications [3, 18, 21, 36, 37]. The numerical tool is written in the Fortran 90 standards coupled with message passing interface (MPI) features [8]. The HDF5 [9, 10] and CGNS libraries [19, 26, 27, 30] are included into the numerical solver in

Technical Editor: André Cavalieri.

✉ Carlos Junqueira-Junior
junior.junqueira@ensam.eu

[1] DynFluid Laboratory, École Nationale Supérieure d'Arts et Métiers, 75013 Paris, France

[2] Instituto de Aeronáutica e Espaço, DCTA/IAE/ALA, São José dos Campos, SP 12228-904, Brazil

[3] Instituto Tecnológico de Aeronáutica, DCTA/ITA, São José dos Campos, SP 12228-900, Brazil

[4] Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Rua Mendeleyev, 200, Campinas, SP 13083-860, Brazil

[5] LatAm Experian DataLab, São Paulo, SP, Brazil

order to implement a hierarchical data format (HDF) and to perform input/output operations efficiently.

Large eddy simulations require a significant amount of computational resources to provide high-fidelity results. The scientific community has been using up to hundreds of million degrees of freedom on simulations of turbulent flow configurations [6, 12, 18, 32]. Researchers and engineers need to be certain that calculations are run with maximum parallel efficiency when allocating computational resources because the access to supercomputers is often restricted and limited. Therefore, it is of major importance to perform scalability studies, regarding an optimal choice of computational load and resources, before running simulations on supercomputers.

The present work addresses the computational performance evaluation of the code when using an Hewlett Packard Enterprise (HPE) cluster from a computational center [5]. The high-performance computing (HPC) solution provides a maximum theoretical peak performance of 127.4 TFLOPS using CPUs, NVIDIA GPUs and Xeon Phi accelerators. Simulations of a pre-defined perfectly expanded jet flow condition are performed using different loads and resources in order to study the strong scalability of the solver. More specifically, nine mesh configurations are investigated running on up to 400 processors in parallel. The number of degrees of freedom starts with 5.8 million points and scales to 1.0 billion points. The speedup and computational efficiency curves are measured for each grid configuration.

The supercomputer is described after the introduction section. Then, numerical formulations and the implementation aspects of the tool are discussed. In the sequence, the strong scalability results are presented to the reader followed by the discussion on the validation of the solver. In the end, the one can find the concluding remarks and acknowledgements.

2 Computational resources

The current work is included in the CEPID-CeMEAI [5] project from the Applied Mathematics department of the University of São Paulo. This project is addressed to four main research subjects: optimization and operational research; computational intelligence and software engineering; computational fluid mechanics; and risk evaluation. The CEPID-CeMEAI provides access to a high-performance computer server located at the University of São Paulo, named Euler. The system presents a maximum theoretical peak performance of 127.4 TFLOPS using a hybrid parallel processing architecture which has 144 CPU nodes, 4 fat nodes, 6 GPU nodes and 1 Xeon Phi node with a total of 3428 computational cores. The detailed description of each node configuration is presented in Table 1. Two storage systems are available with 175Tb each one, the network file system (NFS) and the *Lustre*® file system [23]. The network communication is performed using InfiniBand and Gigabit Ethernet. Red Hat Enterprise Linux [28] is the operating system of the cluster, and Altair PBS Pro [1] is the job scheduler available.

Table 1 Computer cluster configuration

Nodes	Processor	Memory	Nb. cores
104	2 × CPU *Intel*® *Xeon*® E5-2680v2	128GB DDR3	20
40	2 × CPU *Intel*® *Xeon*® E5-2680v4	128GB DDR3	28
4	2 × CPU *Intel*® *Xeon*® E5-2667v4	512GB DDR3	16
6	2 × CPU *Intel*® *Xeon*® E5-2650v4	128GB DDR3	24
	+ 1 × GPU *Nvidia*® *Tesla*® P-100	16GB DDR3	3584
1	2 × CPU *Intel*® *Xeon*® E5-2680v2	128GB DDR3	20
	+ 1 × *Intel*® *Xeon Phi*™ C2108-RP2	8GB DDR3	60

3 Large eddy simulation formulation

The numerical simulations of supersonic jet flow configurations are performed based on the large eddy simulation formulation [11]. This set of equations is based on the principle of scale separation over the governing equations used to represent the fluid dynamics, the Navier–Stokes formulation. A filtering procedure can be used to describe the scale separation in a mathematical formalism. The idea is to model the small turbulent structures and to calculate the bigger ones. Subgrid scale (SGS) closures are added to the filtered equations in order to model the effects of small-scale turbulent structures. The Navier–Stokes equations are written in the current work using the filtering procedure of Vreman [35] as

$$
\begin{aligned}
&\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u}_j\right) = 0\,,\\
&\frac{\partial}{\partial t}\left(\overline{\rho}\widetilde{u}_i\right) + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u}_i\widetilde{u}_j\right) + \frac{\partial \overline{p}}{\partial x_i} - \frac{\partial \tau_{ij}}{\partial x_j}\\
&\qquad + \frac{1}{3}\frac{\partial}{\partial x_j}\left(\delta_{ij}\sigma_{kk}\right) = 0\,,\\
&\frac{\partial \overline{e}}{\partial t} + \frac{\partial}{\partial x_j}\left[\left(\overline{e}+\overline{p}\right)\widetilde{u}_j\right] - \frac{\partial}{\partial x_j}\left(\tau_{ij}\widetilde{u}_i\right)\\
&\qquad + \frac{1}{3}\frac{\partial}{\partial x_j}\left[\left(\delta_{ij}\sigma_{kk}\right)\widetilde{u}_i\right] + \frac{\partial q_j}{\partial x_j} = 0\,,
\end{aligned}
\qquad (1)
$$

in which t and x_i are independent variables representing time and spatial coordinates of a Cartesian coordinate system,

$\mathbf{x}$, respectively. The components of the velocity vector, $\mathbf{u}$, are written as u_i and $i = 1, 2, 3$. Density, pressure and total energy per unit volume are written as ρ, p and e, respectively. The $(\bar{\cdot})$ and $(\tilde{\cdot})$ operators are used in order to represent filtered and Favre-averaged properties, respectively.

It is important to remark that the System I filtering procedure [35] neglects the double correlation term, $\widetilde{u_i u_j}$, which is present into the total energy per unit volume equation in order to write

$$\bar{e} = \frac{\bar{p}}{\gamma - 1} + \frac{1}{2}\bar{\rho}\tilde{u}_i\tilde{u}_i \,. \tag{2}$$

The heat flux, q_j, is given by

$$q_j = \left(\kappa + \kappa_{sgs}\right)\frac{\partial \tilde{T}}{\partial x_j} \,. \tag{3}$$

where T and κ stand for static temperature and the thermal conductivity coefficient, respectively. The last can be expressed as

$$\kappa = \frac{\mu C_p}{Pr} \,, \tag{4}$$

in which Cp is the specific heat at constant pressure, μ is the dynamic viscosity coefficient and Pr is the Prandtl number, which is equal to 0.72 for air. The SGS thermal conductivity coefficient, κ_{sgs}, is written as

$$\kappa_{sgs} = \frac{\mu_{sgs} C_p}{Pr_{sgs}} \,, \tag{5}$$

where Pr_{sgs} is the SGS Prandtl number, which is equal to 0.9 for static SGS closures and μ_{sgs} is the eddy viscosity coefficient that is calculated by the SGS model. In the present work, the dynamic viscosity coefficient is calculated using the Sutherland Law which is given by

$$\mu\left(\tilde{T}\right) = \mu_\infty \left(\frac{\tilde{T}}{\tilde{T}_\infty}\right)^{\frac{3}{2}} \frac{\tilde{T}_0 + S_1}{\tilde{T} + S_1}, \text{ with } S_1 = 110.4K \,. \tag{6}$$

One can use an equation of state to correlate density, static pressure and static temperature,

$$\bar{p} = \bar{\rho} R \tilde{T} \,, \tag{7}$$

in which R is the gas constant, written as

$$R = C_p - C_v \,, \tag{8}$$

and C_v is the specific heat at constant volume. The shear-stress tensor, τ_{ij}, is written as,

$$\tau_{ij} = 2\left(\mu + \mu_{sgs}\right)\left(\tilde{S}_{ij} - \frac{1}{3}\delta_{ij}\tilde{S}_{kk}\right) \tag{9}$$

where the components of the rate-of-strain tensor, $\tilde{S}_{ij}$, are given by

$$\tilde{S}_{ij} = \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) \,. \tag{10}$$

The SGS stress tensor components can be written using the eddy viscosity coefficient [31],

$$\sigma_{ij} = -2\mu_{sgs}\left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\right) + \frac{1}{3}\delta_{ij}\sigma_{kk} \,. \tag{11}$$

In the present article, the eddy viscosity coefficient, μ_{sgs}, and the components of the isotropic part of the SGS stress tensor, σ_{kk}, are not considered for the calculations. Previous validation results performed by the authors [17, 18] indicate that the characteristics of numerical discretization of JAZzY can overcome the effects of subgrid scale models. The same conclusion can be found in the work of Li and Wang [20]. Therefore, one can write the large eddy simulation set of equations can be written in a more compact form as

$$\frac{\partial \mathbf{Q}}{\partial t} = -\mathbf{RHS} \,, \tag{12}$$

where $\mathbf{Q}$ stands for the conservative properties vector, given by

$$\mathbf{Q} = \left[\bar{\rho}\,, \bar{\rho}\tilde{u}_i\,, \bar{e}\right]^T \,, \tag{13}$$

and $\mathbf{RHS}$, which stands for the right-hand side of equation, represents the contribution of inviscid and viscous fluxes terms from Eq. 1. The components of the right-hand-side vector are written as

$$\mathbf{RHS}_i = \frac{\partial E_i}{\partial x_j} - \frac{\partial F_i}{\partial x_j} \,, \tag{14}$$

in which E_i and F_i are the components of inviscid and viscous flux vectors, respectively, given by

$$\mathbf{E} = \begin{bmatrix} \bar{\rho}\tilde{u}_j \\ \bar{\rho}\tilde{u}_i\tilde{u}_j + \bar{p}\delta_{ij} \\ \left[\left(\bar{e} + \bar{p}\right)\tilde{u}_j\right] \end{bmatrix} \text{ and } \mathbf{F} = \begin{bmatrix} 0 \\ \tau_{ij} \\ \tau_{ij}\tilde{u}_i - q_j \end{bmatrix} . \tag{15}$$

Spatial derivatives are calculated in a structured finite-difference context and the formulation is re-written for the general curvilinear coordinate system [17]. The numerical flux is computed through a central difference scheme with the explicit addition of anisotropic scalar artificial dissipation model of Turkel and Vatsa [34]. The time-marching method is an explicit 5-stage Runge–Kutta scheme developed by Jameson et al. [16].

Boundary conditions for the LES formulation are imposed in order to represent a supersonic jet flow into a 3-D computational domain with cylindrical shape. Figure 1 presents a lateral view and a frontal view of the computational domain used in the present work and the positioning of the entrance, exit, centerline, far field, and periodic boundary conditions.

A flat-hat velocity profile is implemented at the entrance boundary condition through the use of the 1-D characteristic relations for the 3-D Euler equations in order to create a jet-like flow configuration. The set of properties, then, determined is computed from within and from outside the computational domain. Riemann invariants [22] are used in order to calculate properties at the far-field surfaces where the normal-to-face components of the velocity are computed by a zero-order extrapolation from inside the computational domain. The angle of flow at the far-field boundary is assumed fixed. The remaining properties are obtained as a function of the jet Mach number, which is a known variable.

The flow configuration is assumed to be subsonic at the exit plane of the domain. Therefore, the pressure is obtained from the outside, i.e., it is assumed given the internal energy and components of the velocity are calculated by a zero-order extrapolation from the interior of the domain. Then, density, ρ, and total energy per unit volume, e, are computed at the exit boundary using the extrapolated properties and the imposed pressure at the output plane. The first and last points in the azimuthal direction are superposed in order to close the 3-D computation domain and create a periodicity boundary condition. An adequate treatment of the centerline boundary is necessary since it is a singularity of the coordinate transformation. The conserved properties are extrapolated from the adjacent longitudinal plane and averaged in the azimuthal direction in order to define the updated properties at the centerline of the jet. Furthermore, the fourth-difference terms of the artificial dissipation scheme of Turkel and Vatsa [34] are carefully treated in order to avoid five-point difference stencils at the centerline singularity.

The reader can find further details about the spatial discretization, time-marching scheme and implementation of boundary conditions in the work of Junqueira-Junior [17] and Junqueira-Junior et al. [18] which present the validation of the large eddy simulation solver.

4 Parallel implementation aspects

The solver is developed to calculate the LES set of equations, Eq. 12, for supersonic jet flow configurations using the Fortran 90 standard. The spatial discretization of the formulation is based on a centered finite-difference approach with the explicit addition of anisotropic scalar artificial dissipation model of Turkel and Vatsa [34], and the time integration is performed using an explicit second-order 5-stage Runge–Kutta scheme [16].

Parallelism is achieved using the single program multiple data, SPMD, approach [7] and the exchange of messages provided by MPI protocols [8]. The algorithm of the LES solver is structured in two main steps. Firstly, a preprocessing routine reads a mesh file and performs a balanced partitioning of the domain procedure. Then, in the processing routine, each MPI rank reads its correspondent grid file and starts the calculations.

The preprocessing routine is run separately from the processing step. It reads an input file with the partitioning configuration and a 2-D grid file. Next, the preprocessing code calculates the number of points in the axial and azimuthal directions in order to perform the partitioning and the extrusion in the third direction for each sub-domain. The segmentation of the grid points is illustrated in Fig. 2a. A matrix index notation is used in order to represent positions of each partition where NPX and NPZ denote the number of partitions in the axial and azimuthal directions, respectively.

In the case of a non-exact domain division, the remaining points are spread among the partitions in order to have a well-balanced task distribution. Algorithm 1 presents the details of this division, and Fig. 2b illustrates the balancing procedure in one direction, where *LocNbPt*, *TotNbPt*, *NbPart* and *PartIndex* stand for local number of points in one direction, total number of points in one direction, number of partitions in one direction, and the index of the partitions in one direction. The same algorithm is used to perform the partitioning procedure in both axial and azimuthal directions. This preprocessing part of code is executed sequentially.

Algorithm 1: Optimized partitioning approach.

```
1 begin
2     int LocNbPt // Loc. nb. of pt. in one dir.;
3     int TotNbPt // Tot. nb. of pt. in one dir.;
4     int NbPart // Nb. of part. in one dir.;
5     int PartIndex // Part. index in one dir.;
6     LocNbPt = (TotNbPt/NbPart) ;
7     if [PartIndex < mod(TotNbPt,NbPart)] then
8         LocNbPt = LocNbPt + 1 ;
```

The mesh files, for each domain, are written after the optimized partitioning procedure using the CFD General Notation System (CGNS) standard [19, 26, 27, 30]. This standard is based on the HDF5 [9, 10] libraries which provide tools for hierarchical data format (HDF) and can perform input/output operations efficiently. The authors have chosen to write one CGNS grid file for each partition in order to have each MPI rank performing I/O operations independently, i.e., in parallel, during the processing step of the calculations. Moreover, each MPI rank can also write its own time-dependent solution to a local CGNS file. Such an approach avoids synchronizations during checkpoints, which can be a significant drawback in HPC applications.

3 and 4 of the same algorithm, each rank reads a local-domain CGNS file and calculates Jacobian and metric terms, which are used for the general curvilinear coordinates transformation. Ghost points are added to the boundaries of local mesh at the axial and azimuthal directions in order to carry information of neighbor partition points,

Algorithm 2: Implementation of large eddy simulation formulation.

```
1  begin
2      Read(Input) ;                                  // Read flow conf.
3      Read(Mesh) ;                                   // Read local mesh
4      Calc(Jacob.,Metric) ;
5      Create(Ghost pts.) ;
6      if Restart then
7          Read(Rest. data);                          // Read check-point sol.
8      else
9          Calc(I.C.) ;                               // Impose I.C.
10     Comm(Jacob.,Metric,I.C.);
11     while Nb. it. do                               // Main It. loop
12         for $\ell \leftarrow 1$ to 5 do                       // 5-steps Runge-Kutta
13             MPI Sync. ;                            // MPI Barrier Func.
14             for $i,j,k$ do                         // 3-D spatial loop
15                 Calc(Inv. Flux);                   // Calc. of inviscid vec.
16                 Calc(Art. Diss.);                  // Calc. of artficial dissip. terms
17                 Calc(Visc. Flux);                  // Calc. of viscous vec.
18                 Calc(RHS Vec.);                    // Calc. of RHS vec.
19                 Update(Cons. Vec.,$\alpha_\ell$);            // $\ell$-th R-K step
20             Update(B.C.,Visc.);                    // Update B.C. and viscosity coef.
21             Comm(Cons. Vec..);                     // Asynchronous MPI comm.
22         Write(Output);                             // Writes time-dependent CGNS sol.
```

After the preprocessing routine, the solver can start the simulation. A brief overview of the computing part of the LES code is presented in Algorithm 2. Primarily, every MPI process reads the same ASCII file with input data such as flow configurations and simulation settings, as indicated in line 2 of Algorithm 2. In the sequence, lines line 5 in Algorithm 2. The artificial dissipation scheme of Turkel and Vatsa [34] implemented in the code [15] uses a five points stencil which requires information of the two neighbors of a given mesh point. Hence, two-layer ghost points are created at the beginning and at the end of each partition. Figure 3 presents the layer of ghost points used

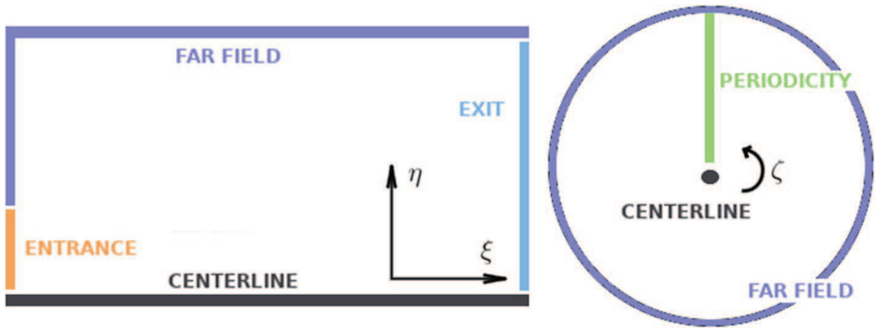


Fig. 1 Two-dimensional lateral and frontal illustrations of the domain which indicate the positioning of boundary conditions

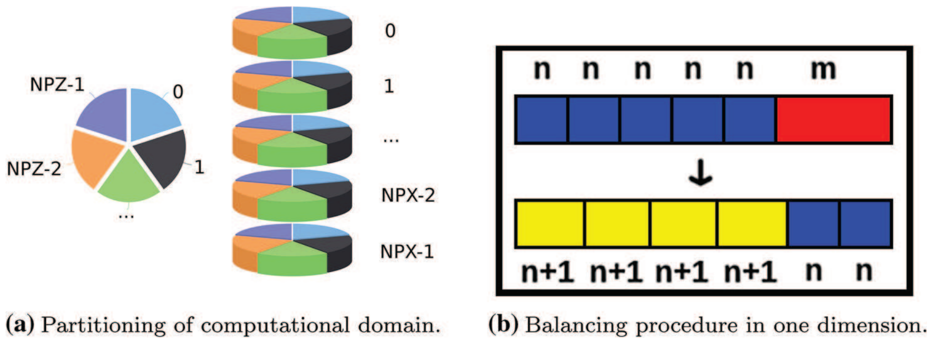

NPZ-1
0
NPZ-2
1
...
0
1
...
NPX-2
NPX-1
(a) Partitioning of computational domain.
n n n n n m
n+1 n+1 n+1 n+1 n n
(b) Balancing procedure in one dimension.

Fig. 2 Partitioning and balancing approaches

in the present code. The yellow and black layers represent the axial and azimuthal ghost points, respectively. The green region represents the local partition grid points.

The initial conditions of the flow configurations are imposed following the sequence of tasks of the processing algorithm. They are calculated using data from a previous checkpoint or, from the input data depending on if it is a restart simulation or not, as presented in lines 5 to 9 of Algorithm 2. An asynchronous MPI communication of metric terms, Jacobian terms, and conservative properties, set as initial conditions, is performed between partitions which share neighbor surfaces in order to update all ghost points before starting the time integration.

AQ2 The core of the code consists of the while loop indicated in line 11 of Algorithm 2. This specific part of the code is evaluated in the scalability study presented in Sect. 5, *Computational Performance Study*. This loop performs the Runge–Kutta time integration iteratively, for all grid points of the computational domain, until reaching the requested number of iterations. A succession of computing subroutines is performed at each call for the time-marching scheme. It starts with synchronization of MPI ranks in order to avoid race conditions followed by the calculation of the inviscid flux vectors, artificial dissipation terms, viscous flux vector and right-hand side vector, chronologically. The boundary condition and the dynamic viscosity coefficient are updated at the end of each time-marching step followed by a non-blocking communication of the conservative properties at the partition boundaries. If necessary, a time-dependent solution is incremented to the CGNS output le at the end of the main loop iteration.

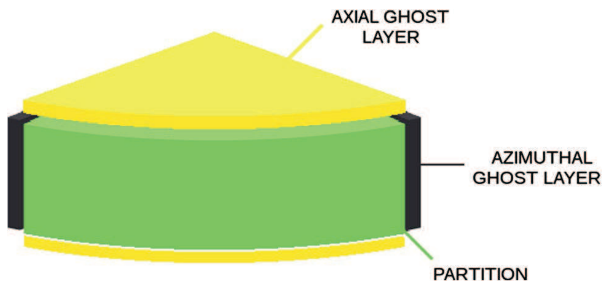


Fig. 3 The positioning of ghost layer points

5 Computational performance study

The numerical discretization approach used in the present article requires very refined grids in order to reproduce high-fidelity results for the simulation of supersonic jet flows configurations. Therefore, parallel computing with efficient inter-partition data exchanges is mandatory. The parallel efficiency of the code is measured using different computational loads, and the results are presented in the present section. The calculations performed in the current article

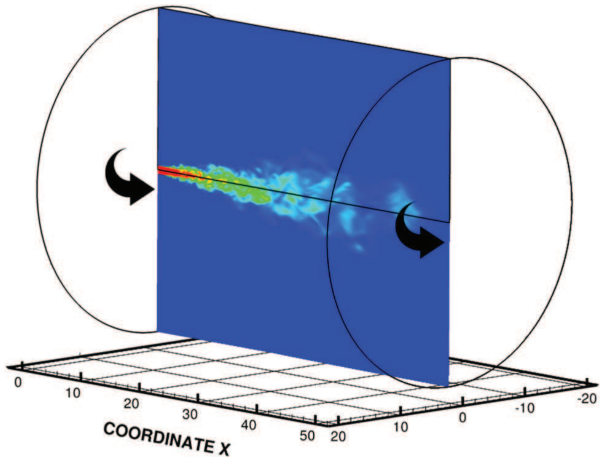


Fig. 4 A two-dimensional surface, colored by the velocity magnitude contours, extracted from the complete geometry used on the strong scalability study

Table 2 Configuration of computational meshes used in the current study

Mesh	No. Pt. Axial Dir.	No. Pt. Radial Dir.	No. Pt.
A	128	128	5.9M
B	256	128	11.8M
C	256	256	23.7M
D	512	256	47.3M
E	512	512	94.6M
F	1024	512	189.3M
G	1024	1024	378.5M
H	2048	1024	757.1M
I	1700	1700	1.0B

are run using the 104 nodes of the Euler supercomputer with the *Intel*® *Xeon*® E5-2680v2 architecture and 128GB DDR3 rapid access memory.

The *Intel*® Composer XE compiler, version 15.0.2.164, is used in the present work. A set of compiling flags which have been tested in previous work [17] are used in the present paper:

–O3 –xHost –ipo –no-prec-div –assume-buffered –override-limits

where **O3** enables aggressive optimization such as global code scheduling, software pipelining, predication and speculation, prefetching, scalar replacement and loop transformations; **xHost** tells the compiler to generate instructions for the highest instruction set available on the compilation host processor; **ipo** uses automatic, multi-step process that allows the compiler to analyze the code and determine where you can benefit from specific optimizations; **no-prec-div** enables optimizations that give slightly less precise results than full division; **assume-buffered_io** tells the compiler to accumulate records in a buffer; and **override-limits** deals with very large, complex functions and loops.

5.1 Scalability setup

Simulations of an unheated perfectly expanded jet flow are performed using different grid sizes and number of processors in order to study the strong scalability of JAZzY. The jet entrance Mach number is 1.4. The pressure ratio, PR $= P_j/P_\infty$, and the temperature ratio, TR $= T_j/T_\infty$, between the jet entrance and the ambient freestream conditions, are equal to one, i.e., PR = 1 and TR = 1 where the j subscript identifies the properties at the jet entrance and the ∞ subscript stands for properties at the far-field region. The Reynolds number of the jet is Re $= 1.57 \times 10^6$, based on the jet entrance diameter, D. The time increment, Δt, used for the validation study is 1×10^{-4} dimensionless time units. A stagnated flow is used as the initial condition for the simulations.

The same geometry is used for the computational evaluation, where the 2-D surface of this computational domain, as presented in Fig. 1, is 30 dimensionless length and 10

Table 3 Number of partitions in the azimuthal direction for a given number of processors

Nb. Proc.	Nb. of Part. in the Azimuthal Dir.							
1	1							
2	1	2						
5	1	5						
10	1	2	5	10				
20	1	2	4	5	10	20		
40	1	2	4	5	8	10	20	40
80	2	4	5	8	10	20	40	
100	2	4	5	10	20	25		
200	4	8	10	20	25	50		
400	8	16	20	25	50			

dimensionless height. Figure 4 illustrates a 2-D cut of the geometry colored by velocity contours.

The present work uses nine different mesh configurations whose total number of points doubles every time. Table 2 presents the details of each grid design where the first column presents the name of the mesh, while the second and third columns present the number of points in the axial and radial directions, respectively. The last column indicates the total number of points of the mesh. The grid point distribution in the azimuthal direction is fixed at 361. The smallest grid, named as Mesh A, has approximately 5.9 million points, while the biggest grid presents approximately 1.0 billion points.

The solver is able to have different partitioning configurations for a fixed number of sub-domains since the division of the mesh is performed in the axial and azimuthal directions. Therefore, different partitioning configurations are evaluated for a given number of processors. Each simulation performs 1000 iterations or 24 h of computation, and the average of the CPU time per iteration of the while loop indicated in Algorithm 2 is measured in order to calculate the speedups of the solver at the Euler supercomputer. The partitioning configurations which provides the fastest calculation are used to evaluate the performance of the solver. Table 3 presents the number of partitions in the azimuthal direction for each number of processors used to study the scalability of the solver. The first column stands for the number of computational cores, while the second column represents the number of zones in the azimuthal direction used to evaluate the effects of the partitioning on the computation. Calculations performed in the present study are run using one single core up to 400 MPI processes.

A sequential computation is the ideal starting point, s, for a strong scalability study. However, frequently, the computational problem cannot be allocated in one single cluster node due to hardware limitations of the system. Therefore, it is necessary to shift the starting point to a minimum number of resources in which the code can be run. The LES solver has shown to be capable of allocating the five smallest grids, from mesh A to mesh E, in one single node of the Euler computer. Aforementioned indicates that it is possible to run a simulation with 94.6 million grid points using 128GB of RAM. The starting points of mesh F and mesh G are 40 cores, allocated in two nodes, and 80 cores, allocated in four nodes, respectively. Meshes H and I start the strong scalability study using 200 cores in 10 nodes of the Euler computer. Table 4 presents the minimum number of computational cores used by each mesh configuration for the strong scaling study.

Table 4 Scalability starting point for each mesh configuration

Mesh	s
A–E	1
F	40
G	80
H–I	200

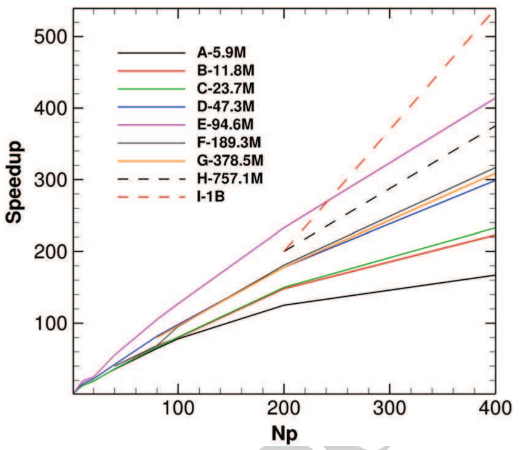


Fig. 5 Speedup curve of the LES solver for different mesh sizes

5.2 Strong scalability study

The speedup, Sp, is used in the present work to measure the strong scaling of the solver and compare it with the ideal theoretical case. There are different approaches used by the scientific community to calculate the speedup [14, 33], which is written in the current article as

$$Sp(m, N) = \frac{T(m, s)}{T(m, N)} . \tag{16}$$

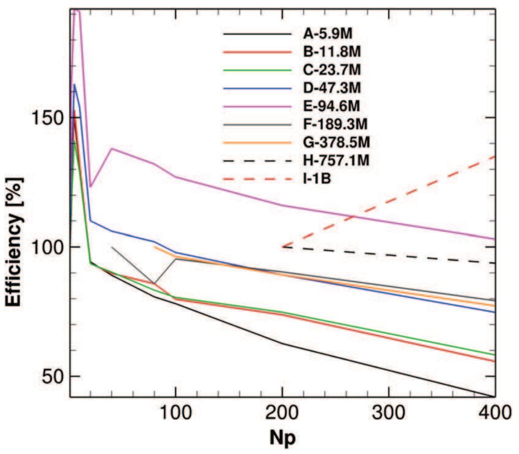


Fig. 6 Parallel efficiency of the LES solver for different mesh sizes

in which T stands for the time spent by mesh m to perform one thousand iterations, N represents the number of computational cores and s is the starting point of the scalability study. The strong scaling efficiency of a given mesh configuration, as a function of the number of processors, is written considering the law of Amdahl [2] as

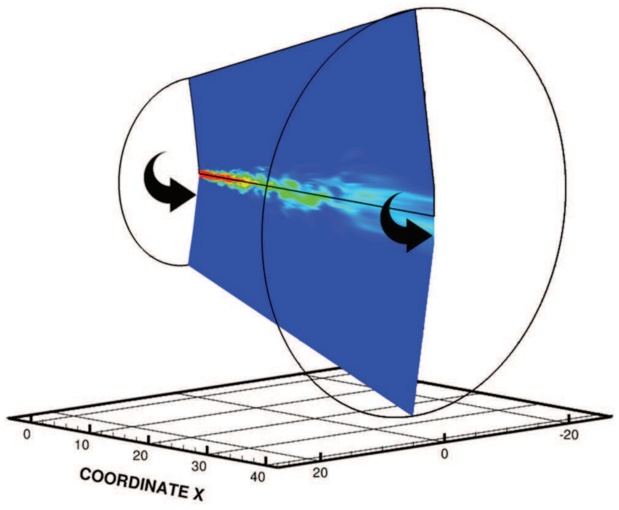
$$\eta(m, N) = \frac{Sp(m, N)}{N} \, . \tag{17}$$

More than 300 calculations are performed when considering all the partitioning configurations and different meshes evaluated in the present paper. The averaged time per iteration is calculated for all numerical simulations in order to study the scalability of the solver. The evolution of speedup and efficiency, as a function of the number of processors, for the nine grids used in the current work is presented in Figs. 5 and 6. The investigation indicates a good scalability of the code. Meshes with more than 50 million points present efficiency bigger than 75% when running with 400 computing cores in parallel. Moreover, mesh E, which has $\approx$ 95 million degrees of freedom, presented an efficiency which equivalent to the ideal case, $\approx$ 100%, when using the maximum number of resources evaluated. One can notice a superlinear scalability for the cases evaluated in the present article. This behavior can be explained by the fact that cache memory can be accessed more efficiently when increasing the total number of processors for a given grid configuration since more computational resources for the same load mean less cache miss [13, 29].

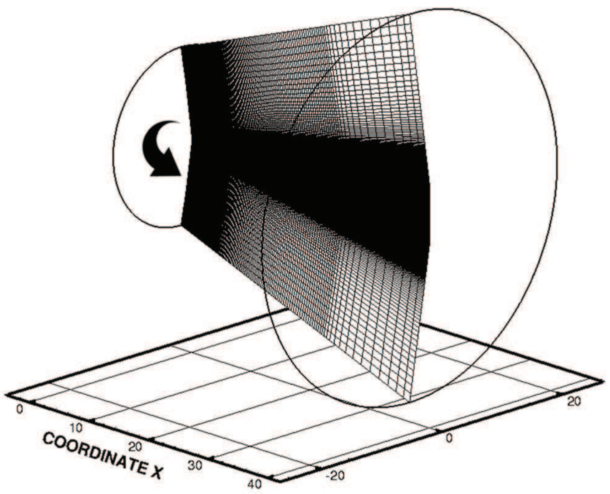
Increasing the size of a computational problem can generate a better scalability study. The time spent with computation becomes more significant when compared to the time spent with communication with the growth of a problem. One can notice such effect for meshes A, B, C, D and E. The speedup and the efficiency increase with the growth of the mesh size. However, such scalability improvement does not happen from mesh E to meshes F, G, H and I. This behavior is originated because the reference used to calculate speedup and efficiency is not the same for all grid configurations. The studies performed using meshes F, G, H and I does not use the serial computation as a reference, which is not the case of calculations performed using meshes A, B, C, D and E.

6 Compressible jet flow simulation

This section presents a compilation of results achieved from the simulation of a supersonic jet flow configuration. This calculation was performed in order to validate the LES code, and it is included here simply to demonstrate that the numerical tool is indeed capable of presenting physically sound results for the problem of interest. Results are compared to numerical [24, 25] and to experimental data [4]. The details of this particular simulation are published in the work of Junqueira et al. [18].



(a) A Two-dimensional surface, colored by velocity magnitude contours, extracted from the full geometry.



(b) A Two-dimensional surface extracted from the full domain superimposed by grid points distribution.

Fig. 7 Illustration of geometry and mesh used in the validation of the LES solver

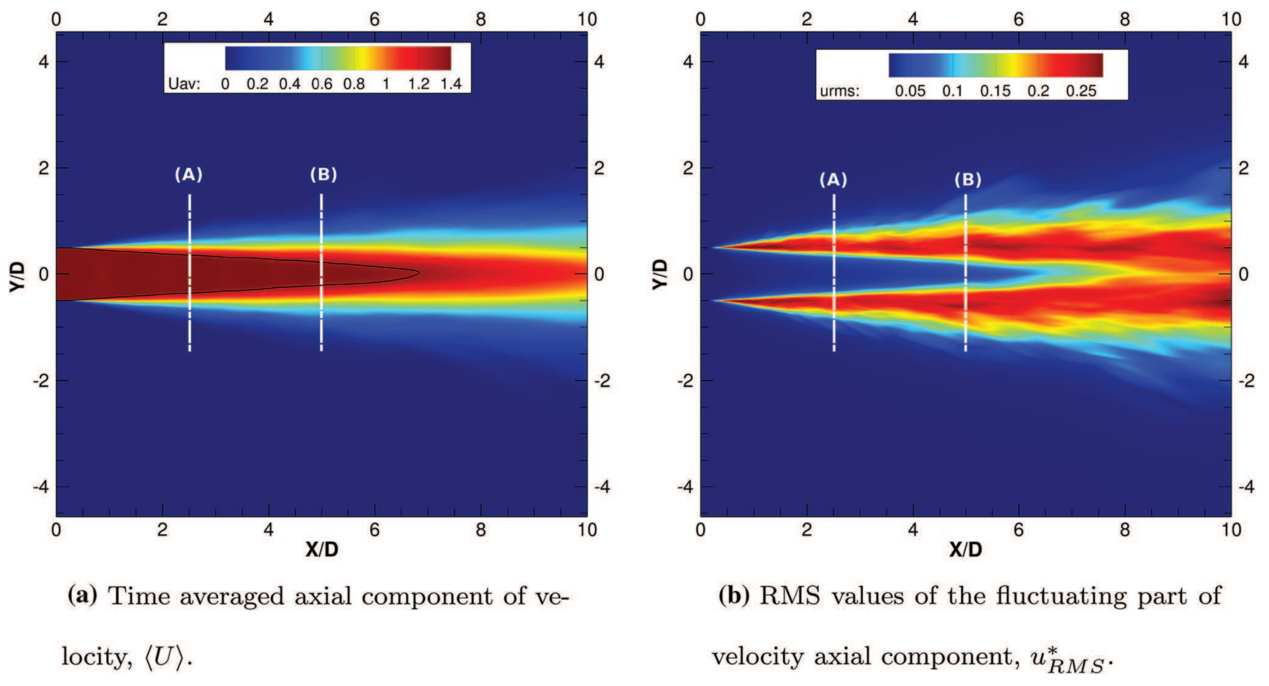


Fig. 8 Lateral view of distributions of $\langle U \rangle$ and u^*_{RMS}. The white dashed lines indicate the positioning of radial cuts where data are extracted and averaged. The solid black line in **a** represents the potential core of the jet

A geometry is created using a divergent shape whose axis length is 40 times the jet entrance diameter, D. Figure 7 illustrates a 2-D cut of the geometry and the grid point distribution used on the validation of the solver. The mesh presents approximately 50 million points. The calculation is performed using 500 computational cores.

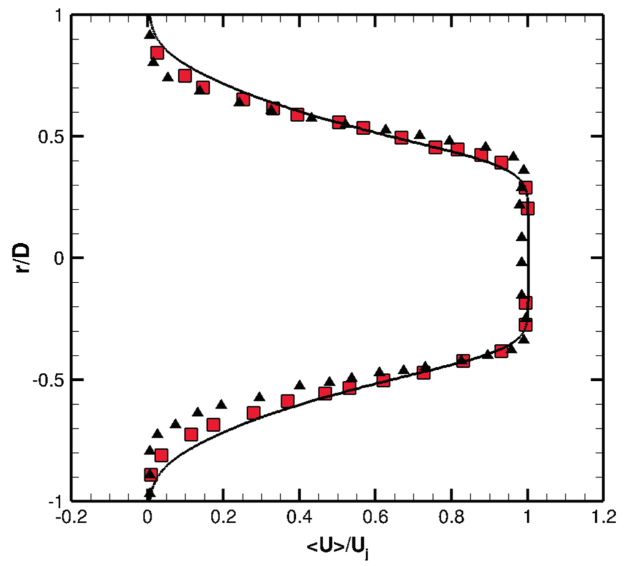
An unheated perfectly expanded jet flow is studied for the present validation. The jet entrance Mach number is 1.4. The pressure ratio, PR= P_j/P_∞, and the temperature ratio, TR= T_j/T_∞, between the jet entrance and the ambient freestream conditions, are equal to one, i.e., PR= 1 and TR= 1. The j subscript identifies the properties at the jet entrance, and the ∞ subscript stands for properties at the far-field region. The Reynolds number of the jet is Re= 1.57×10^6, based on the jet entrance diameter, D. The time increment, Δt, used for the validation study is 1×10^{-4} dimensionless time units.

The boundary conditions previously presented in the *Large Eddy Simulation Formulation* section are applied in the current simulation. The stagnation state of the flow is set as an initial condition of the computation. The calculation runs a predetermined period of time until reaching the statistically steady flow condition. This first pre-simulation is important to assure that the jet flow is fully developed and turbulent. Computations are restarted and run for another period after achieving the statistically stationary state flow. Hence, data are extracted and recorded in a predetermined frequency. Figure 8 indicates the positioning of the two surfaces, (A) and (B), where data are extracted and averaged through time. Cuts (A) and (B) are radial profiles at 2.5D and 5.0D units downstream of the jet entrance. Flow quantities are also averaged in the azimuthal direction when the radial profiles are calculated.

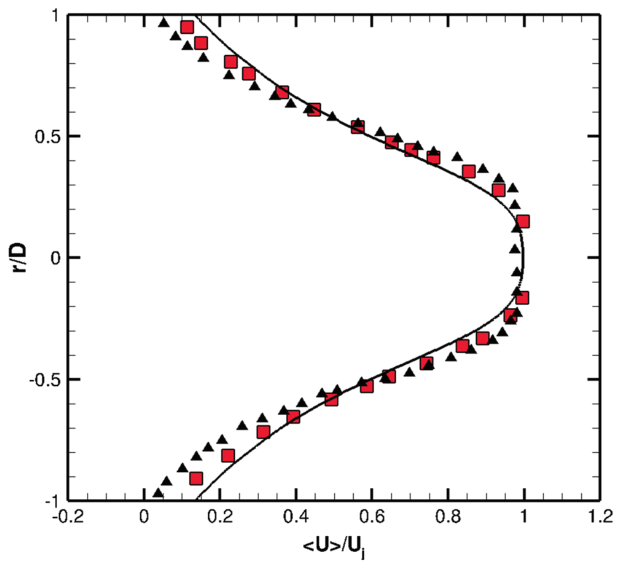
Figure 8a and b presents distributions of time-averaged axial component of velocity and root-mean-square values of the fluctuating part of the axial component of velocity, which are represented in the present work as $\langle U \rangle$ and u^*_{RMS}, respectively. The solid black line indicated in Fig. 8a represents the potential core of the jet, which is defined as the region where the time-averaged axial velocity component is at least 95% of the velocity of the jet at the inlet.

Dimensionless profiles of $\langle U \rangle$ and u^*_{RMS} at the cuts along the mainstream direction of the computational domain are compared with numerical and experimental results in Figs. 9 and 10, respectively. The solid line stands for results achieved using the JAZzY code, while square and triangular symbols represent numerical [24, 25] and experimental [4] data, respectively.

The averaged profiles obtained in the present work correlate well with the reference data at the two positions compared here. It is important to remark that the LES tool can provide good predictions of supersonic jet flow configurations when using a sufficiently fine grid point distribution.

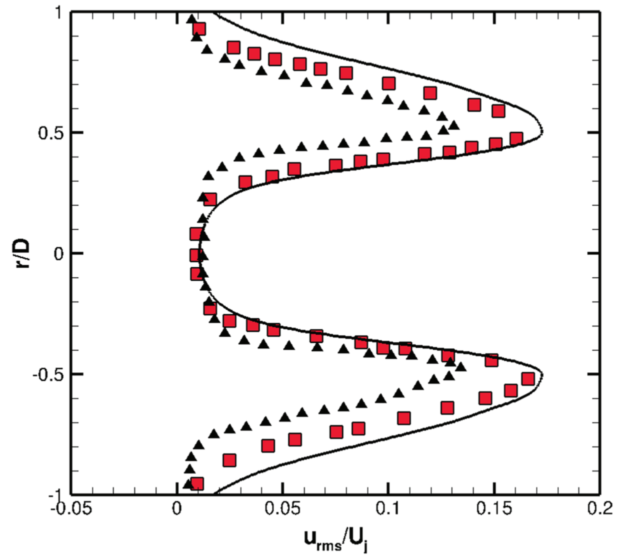


(a) X=2.5D

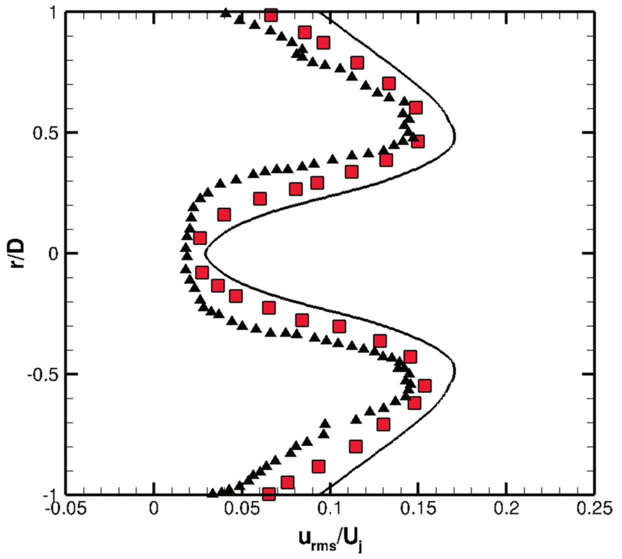


(b) X=5.0D

Fig. 9 Profiles of the averaged axial component of velocity, $\langle U \rangle$, at 2.5D and 5.0D from the entrance: (–) JAZzY results; (🟥) numerical data [24, 25]; (▲) experimental data [4]



(a) X=2.5D



(b) X=5.0D

Fig. 10 Profiles of the RMS of the fluctuation part of axial component of velocity, u^*_{RMS}, at 2.5D and 5.0D from the entrance: (–) JAZzY results; (🟥) numerical data [24, 25]; (▲) experimental data [4]

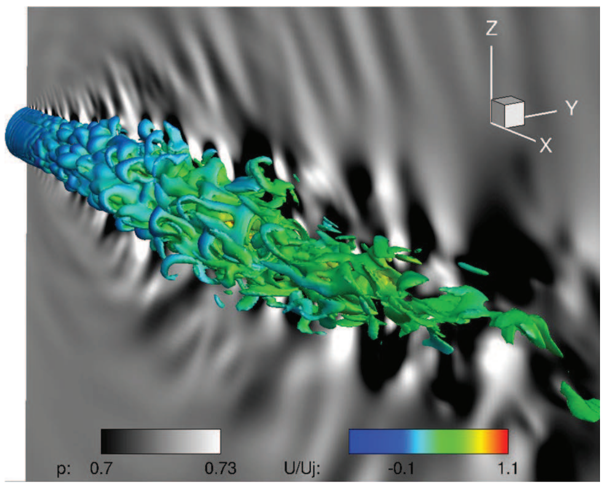


Fig. 11 Instantaneous lateral view of pressure contours, in grayscale, superimposed by 3-D velocity magnitude contours, in color

Therefore, efficient massive parallel computing is mandatory in order to achieve good results.

Figures 11 and 12 present a lateral view of an instantaneous visualization of the pressure contours, in grayscale, superimposed by 3-D velocity magnitude contours and vorti-city magnitude contours, respectively, in color, calculated by the LES tool discussed in the present paper. A detailed visu-alization of the region indicated in yellow, at the jet entrance, is shown in Fig. 12. The resolution of flow features obtained from the jet simulation is more evident in this detailed plot of the jet entrance. One can clearly notice the compression waves generated at the shear layer, and their reflections at the jet axis. Such resolution is important to observe details and behavior of such flow configuration in order to understand the acoustic phenomena which is present in supersonic jet flow configurations.

7 Concluding remarks

The current work is concerned with the performance of a computational fluid dynamics tool for aeroacoustics applications when using a national supercomputer. The HPC system, Euler, from the University of São Paulo presents more than 3000 computational cores and a maximum theoretical peak of 127.4 TFLOPS. The numerical solver is developed by the authors to study supersonic jet flows. Simulations of such flow configurations are expensive and need efficient parallel computing. Therefore, strong scalability studies of the solver are performed on the Euler supercomputer in order to evaluate whether the numerical tool is capable of efficiently using computational resources in parallel.

The computational fluid dynamics solver is developed using the large eddy simulation formulation for perfectly expanded supersonic jet flow. The equations are written using a finite-difference centered spatial discretization with the addition of artificial dissipation. The time integration is performed using a five-step Runge–Kutta scheme. Parallel computing is achieved through non-blocking message pass-ing interface protocols, and inter-partition data are allocated using ghost points. Each MPI partition reads and writes its

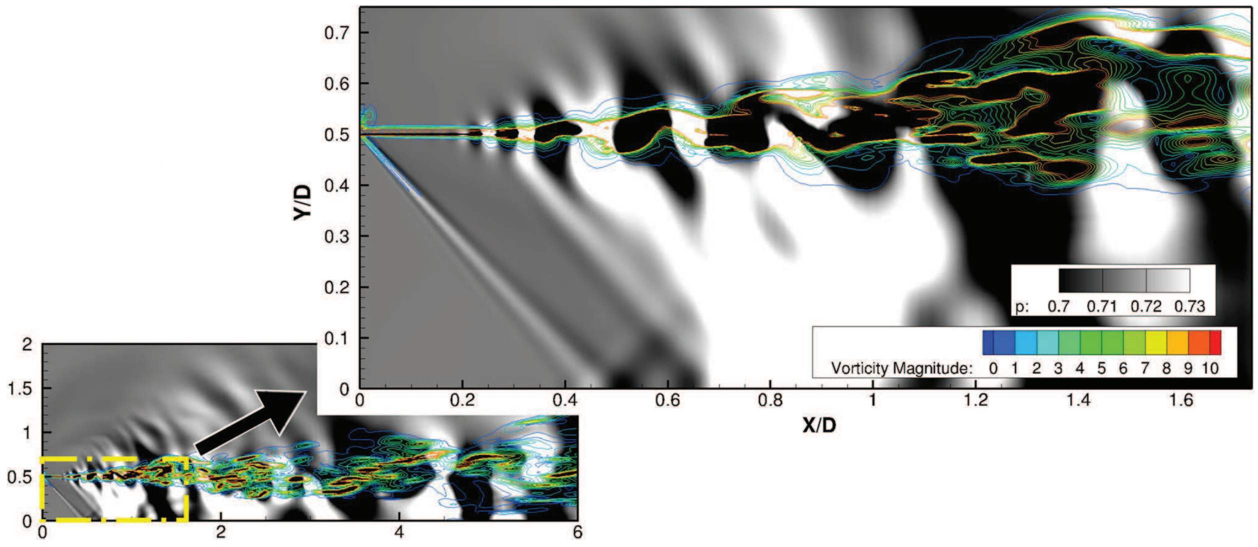


Fig. 12 Lateral and detailed view of pressure contours, in grayscale, superimposed by vorticity magnitude contours, in color. The yellow box indicates the region illustrated in the detailed view

own portion of the mesh that is created on preprocessing routine.

A geometry and a flow condition are defined for the scalability study performed in the present work. Nine point distributions and different partitioning configurations are used in order to evaluate the parallel code under different workloads. The size of the grid configurations starts with 5.9 million points and rises up to approximately 1.0 billion points. Calculations perform 1000 iterations or 24 h of computation using up to 400 cores in parallel. The CPU time per iteration is averaged when the simulation is finished in order to calculate the speedup and scaling efficiency. More than 300 simulations are performed for the scalability study when considering different workloads and partitioning configurations.

The code presented a good scalability for the calculations run in the current paper. The averaged CPU time per iteration decays with the increase in the number of processors in parallel for all computation performed by the large eddy simulation solver evaluated in the present work. Meshes with more than 50 million points indicated an efficiency greater than 75%. The problem with approximately 100 million points presented speedup of 400 and efficiency of 100% when running on 400 computational cores in parallel. Such performance is equivalent to theoretical behavior in parallel. It is important to remark the ability of the parallel solver to treat very dense meshes as the one tested in the present paper with approximately 1.0 billion points. Large eddy simulation demands very refined grids in order to have a good representation of the physical problem of interest. Therefore, it is important to perform simulations of such configuration with a good computation efficiency, and the present scalability study article can all be seen as a guide for future simulations using the same numerical tool on the Euler supercomputer.

Acknowledgements The authors gratefully acknowledge the partial support for this research provided by Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, under the Research Grant Nos. 309985/2013-7, 400844/2014-1 and 443839/2014-0. The authors are also indebted to the partial financial support received from Fundação de Amparo à Pesquisa do Estado de São Paulo, FAPESP, under the Research Grant Nos. 2013/07375-0 and 2013/21535-0.

Compliance with ethical standards

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Altair (2019) PBS works^{TM}, https://www.pbsworks.com/
2. Amdahl GM (1967) Validity of the single processor approach to achieving large scale computing capabilities. In: AFIPS conference proceedings, vol 30. ACM, Atlantic City, N.J., USA, pp. 483–485
3. Bodony D, Lele SK (2005) On using large-eddy simulation for the prediction of noise from cold and heated turbulent jets. Phys Fluids 17(8):085103
4. Bridges J, Wernet MP (2008) Turbulence associated with broadband shock noise in hot jets. In: AIAA Paper No. 2008-2834. In: Proceedings of the 14th AIAA/CEAS aeroacoustics conference (29th AIAA Aeroacoustics Conference), Vancouver, Canada,
5. Center for Mathematical Sciences Applied to Industry (CEPID-CeMEAI) (2016) (CEPID-CeMEAI) web page, https://www.cemeai.icmc.usp.br/
6. Cohen E, Gloerfelt X (2018) Influence of pressure gradients on wall pressure beneath a turbulent boundary layer. J Fluid Mech 838:715–758
7. Darema F (2001) SPMD model: past, present and future. In: 8th European PVM/MPI users' group meeting recent advances in parallel virtual machine and message passing interface, Santorini/Thera, Greece, pp. 23–26
8. Dongarra JJ, Otto SW, Snir M, Walker D (1995) An introduction to the MPI Standard. Technical report, Knoxville, TN, USA
9. Folk M, Cheng A, Yates K (1999) HDF5: a file format and I/O library for high performance computing applications. Proc Supercomput 99:5–33
10. Folk M, Heber G, Koziol Q, Pourmal E, Robinson D (2011) An overview of the HDF5 technology suite and its applications. In: Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases. ACM, pp. 36–47
11. Garnier E, Adams N, Sagaut P (2009) Large eddy simulation for compressible flows. Springer, Berlin
12. Gloerfelt X, Cinnella P (2019) Large eddy simulation requirements for the flow over periodic hills. Flow Turbul Combust 103:55–91 AQ3
13. Gusev M, Ristov S (2014) A superlinear speedup region for matrix multiplication. Concurr Comput Pract Exp 26(11):1847–1868
14. Gustafson JL (1988) Reevaluating Amdahl's law. Commun ACM 31(5):532–533
15. Jameson A, Mavriplis D (1986) Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh. AIAA J 24(4):611–618
16. Jameson A, Schmidt W, Turkel E (1981) Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. In: Proceedings of the AIAA 14th fluid and plasma dynamic conference AIAA paper 81–1259, Palo Alto, California, USA
17. Junqueira-Junior C (2016) Development of a parallel solver for large eddy simulation of supersonic jet flow. PhD thesis, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil,
18. Junqueira-Junior C, Yamouni S, Azevedo JLF, Wolf WR (2018) Influence of different subgrid-scale models in low-order les of supersonic jet flows. J Braz Soc Mech Sci Eng 40(258):1–29
19. Legensky SM, Edwards DE, Bush RH, Poirier DM (2002) CFD general notation system (CGNS)—status and future directions. In: Proceedings of 40th AIAA aerospace sciences meeting & ExhibitAIAA Paper No. 2002-0752, Reno, NV
20. Li Y, Wang ZJ (2015) A priori and a posteriori evaluation of subgrid stress models with the burger's equation. In: Proceedings of 53rd AIAA aerospace sciences meeting AIAA-2015-1283. Kissimmee, Florida, U.S.A, pp. 20
21. Lo SC, Aikens KM, Blaisdell GA, Lyrintzis AS (2012) Numerical investigation of 3-D supersonic jet flows using large-eddy simulation. Int J Aeroacoust 11(7):783–812
22. Long LN, Khan M, Sharp HT (1991) A massively parallel three-dimensional Euler/Navier–Stokes method. AIAA J 29(5):657–666
23. Lustre® (2019) Lustre® filesystem page, https://www.lustre.org/

24. Mendez S, Shoeybi M, Sharma A, Ham FE, Lele S K, Moin P (2010) Large-eddy simulations of perfectly-expanded supersonic jets: quality assessment and validation. In: 48th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition AIAA Paper No. 2010–0271, Orlando, USA
25. Mendez S, Shoeybi M, Sharma A, Ham FE, Leleand SK, Moin P (2012) Large-eddy simulations of perfectly-expanded supersonic jets using an unstructured solver. AIAA J 50(5):1103–1118
26. Poirier D, Enomoto FY (1998) The CGNS system. In Proceedings of 29th AIAA fluid dynamics conference, AIAA Paper No. 98-3007, Albuquerque, NM
27. Poirier DMA, Bush RH, Cosner RR, Rumsey CL, McCarthy DR (2000) Advances in the CGNS database standard for aerodynamics and CFD. In: 38th AIAA Aerospace Sciences Meeting & Exhibit AIAA Paper No. 2000-0681, Reno, NV
28. RedHat (2019) RedHat web page http://www.redhat.com/
29. Ristov S, Prodan R, Gusev M, Skala K (2016) Superlinear speedup in HPC systems: why and when? In: Proceedings of the 2016 federated conference on computer science and information systems. Gdańsk, Poland, pp. 889–898
30. Rumsey CL, Wedan B, Hauser T, Poinot M (2012) Recent updates to the CFD general notation system (CGNS). In: Proceedings of 50th AIAA aerospace sciences meeting, AIAA Paper No. 2012-1264, Nashville, TN, USA, pp. 16
31. Sagaut Pierre (2002) Large eddy simulation for incompressible flows. Springer, Berlin
32. Sciacovelli L, Cinnella P, Gloerfelt X (2017) Direct numerical simulations of supersonic turbulent channel flows of dense gases. J Fluid Mech 821:153–199
33. Sun XH, Chen Y (2010) Reevaluating Amdahl's law in multicore era. J Parallel Distrib Comput 70(2):183–188
34. Turkel E, Vatsa VN (1994) Effect of artificial viscosity on three-dimensional flow solutions. AIAA J 32(1):39–45
35. Vreman AW (1995) Direct and large-eddy simulation of the compressible turbulent mixing layer. PhD thesis, Universiteit Twente
36. Wolf W, Lele SK (2011) Airfoil Aeroacoustics: LES and acoustic analogy predictions. PhD thesis, Stanford, Stanford, CA, USA
37. Wolf WR, Azevedo JLF, Lele SK (2012) Convective effects and the role of quadrupole sources for aerofoil aeroacoustics. J Fluid Mech 708:502–538