On the scalability of CFD tool for supersonic jet flow configurations

Carlos Junqueira-Junior, João Luiz F. Azevedo, Jairo Panetta, William R. Wolf, Sami Yamouni

On the Scalability of CFD Tool for Supersonic Jet Flow Configurations

Carlos Junqueira-Junior[a], João Luiz F. Azevedo[b], Jairo Panetta[c], William R. Wolf[d], Sami Yamouni[e]

[a]*Arts et Métiers Institute of Technology, DynFluid, CNAM, HESAM University, Paris, France*
[b]*Instituto de Aeronáutica e Espaço, São José dos Campos, Brazil*
[c]*Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil*
[d]*Universidade Estadual de Campinas, Campinas, Brazil*
[e]*DataLab Serasa Experian, São Paulo, Brazil*

Abstract

New regulations are imposing noise emissions limitations for the aviation industry which are pushing researchers and engineers to invest efforts in studying the aeroacoustics phenomena. Following this trend, an in-house computational fluid dynamics tool is build 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 grid points. The present work addresses the evaluation of code improvements along with the computational performance of the solver running on a computer with maximum theoretical peak of 2.727 PFlops. Different mesh configurations, whose size varies from a few hundred thousand to approximately one billion grid points, are evaluated in the present paper. Calculations are performed using different workloads in order to assess the strong and weak scalability of the parallel computational tool. Moreover, validation results of a realistic flow condition are also presented in the current work.

Keywords:
Computational Fluid Dynamics, Large Eddy Simulation, Scalability, Supersonic Jet Flow

1. Introduction

Exposure to noise can have a huge impact on health and induce a range of physiological reactions such as an increase in blood pressure, heart rate and breathing. Even sudden noise levels commonly experienced in every day, such as busy streets, can cause damage to the health [1]. High level of noise emissions on residents near airports has forced governments to create laws, regulations and guidelines for the certification of noise-emitting airplanes [2, 3]. According to the new constraints, airplanes which do not respect the noise emissions limitations may not be operated from all airports or their operators must pay additional fees for noise emission. Moreover, noisy aircraft may not be operated during the night. Hence, airlines have to consider the noise-related airport fees in their operating costs which can increase the price of flights. Such a scenario has been pushing the civil aviation industry to invest significant efforts in studying aeronautical noise emissions. More specifically, on the aeroacoustic analogy of supersonic jet flows. Such configuration can represent free-jet engine flows that contribute significantly to the total sound emissions of an airplane [4].

The authors are interested on the study of unsteady property fields of 3-D supersonic jet flow configurations which can provide important information in order to eventually understand the acoustic phenomena. Experimental techniques used to evaluate such flow configuration are complex and require considerably expensive apparatus. Therefore, the authors have developed a numerical tool, JAZzY [5], based on the large eddy simulation (LES) formulation [6] in order to perform time-dependent simulations of supersonic compressible jet flows. The large eddy simulation approach has been successfully used by the scientific community and can provide high fidelity numerical data for aeroacoustic applications [7, 8, 9, 10, 11]. The numerical tool is written in the Fortran 90 standards coupled with Message Passing Interface (MPI) features [12]. The HDF5 [13, 14] and CGNS libraries [15, 16, 17, 18] are included into the numerical solver in order to implement a hierarchical data format (HDF) and to perform Input/Output operations efficiently.

Large eddy simulations of supersonic flows require efficient use of significant amount of computational resources in order to provide trustworthy results at an acceptable cost. Hence, the LES tool is continually improved. The latest code release includes modifications in MPI data exchanges and reading/writing routines. Classical blocking communicators are replaced by asynchronous protocols and the sequential mesh reading is re-written in a parallel fashion. Moreover, the partitioning routine is optimized regarding the optimal workload. The present work addresses the computational performance evaluation of the in-house numerical tool on a Brazilian scientific computer named as Santos Dumont [19]. The HPC system was cited on the top 500 list [20, 21] from 2015 to 2017 with a maximal LINPACK [22, 23] performance achieved of 456.8 TFlops and a theoretical peak performance of 657.5 TFlops. A

recent upgrade brings the supercomputer up to the 193rd position on the top 500 list of November 2019. This new version delivers maximal LINPACK and theoretical peak performances of 1.849 PFlops and 2.727 PFlops, respectively. Strong and weak scalability tests are performed in the present work using the TFlops version of the cluster. Numerical simulations of a supersonic jet flow configuration are addressed as a test case in order to evaluate the JAZzY code using up to approximately three thousand computational cores in parallel for problems with up to one billion grid points.

The present article is structured into an introduction followed by a description of the computer architecture. Then, the numerical formulation used by the JAZzY code and the latest results of supersonic jet flow simulations achieved by the LES solver are presented to the reader. In the sequence, a detailed discussion about the parallel implementation and features of the code is performed followed by the results of the scalability study performed in the Santos Dumont supercomputer. Numerical results achieved during the validation of the numerical tool are also presented here. In the end, the reader can find the concluding remarks section and the acknowledgements.

2. Computer Configuration

Santos Dumont supercomputer was acquired from Atos HPC systems by [19] in 2015 by the National Laboratory for Scientific Computing (LNCC) [24] in the city of Petrópolis, state of Rio de Janeiro, Brazil. The main idea of the laboratory, which is affiliated with the Ministry of Science, Technology, Innovations and Communications (MCTIC) in Brazil is to provide computational resources for research from different areas of study such as Engineering, Astronomy, Biology, Chemistry, Geosciences and Linguistics.

The full HPC system is in the 193rd position of the top 500 list of November 2019 [20] with a maximal LINPACK [22, 23] performance achieved of 1.849 PFlops and a theoretical peak performance of 2.727 PFlops. The scalability of the CFD solver is evaluated using a section of the supercomputer with maximal LINPACK performance and theoretical peak performance of 456.8 TFlops and 657.5 TFlops, respectively. This Teraflop partition presents 18,144 computational cores (CPU) spread among 756 nodes (24 computational cores per node). Graphic processing units (GPU) and Xeon PHI accelerators are also coupled to some of the available computing nodes. Moreover, this partition provides a fat-node with 240 computational cores and 6 TB of rapid access memory. A detailed description of computing nodes configuration is presented in Tab. 1. The computational resource has a *Lustre*® [25] file system with a storage capacity of approximately 1.7 PBytes. There is also a secondary archive system with a storage capacity of approximately 640 TBytes. The 756 nodes of the cluster are interconnected by a infiniband network. Red Hat Enterprise Linux [26] is the operating system of the cluster and Slurm [27] is used as workload manager.

Table 1: Santos Dumont Teraflop partition configuration.

Nodes	Processor	Memory	Nb. Cores
504	2 x CPU *Intel*® *Xeon*® E5-2695v2	64GB	24
198	2 x CPU *Intel*® *Xeon*® E5-2695v2	64GB	24
	+ 2 x GPU *Nvidia*® K-40	12GB	5760
54	2 x CPU *Intel*® *Xeon*® E5-2695v2	64GB	24
	+ 2 x *Intel*® *Xeon Phi*™ 7120	16GB	122
1	16 x CPU *Intel*® Ivy 2.4GHz	6TB	240

3. Large Eddy Simulation Formulation

The numerical simulations of supersonic jet flow configurations are performed based on the large eddy simulation formulation [6]. 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. Such scale separation procedure is addressed as a filtering procedure in a mathematical formalism. The idea is to filter the small turbulent structures and to calculate the bigger ones. The Navier-Stokes equations, using the filtering procedure of Vreman [28], is written in the current work as

$$\begin{aligned} &\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_j}\right) = 0\,, \\ &\frac{\partial}{\partial t}\left(\overline{\rho}\widetilde{u_i}\right) + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u_i}\widetilde{u_j}\right) + \frac{\partial}{\partial x_j}\left(\overline{p}\delta_{ij}\right) - \frac{\partial \tau_{ij}}{\partial x_j} = 0\,, \\ &\frac{\partial \overline{e}}{\partial t} + \frac{\partial}{\partial x_j}\left[\left(\overline{e}+\overline{p}\right)\widetilde{u_j}\right] - \frac{\partial}{\partial x_j}\left(\tau_{ij}\widetilde{u_i}\right) + \frac{\partial q_j}{\partial x_j} = 0\,, \end{aligned} \tag{1}$$

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 mass unit are denoted by ρ, p and e, respectively. The $\overline{(\cdot)}$ and $\widetilde{(\cdot)}$ operators are used in order to represent filtered and Favre averaged properties, respectively. The filtered total energy per mass unit [28] can be written as

$$\overline{e} = \frac{\overline{p}}{\gamma - 1} + \frac{1}{2}\overline{\rho}\widetilde{u_i}\widetilde{u_i}\,. \tag{2}$$

The heat flux, q_j, is written as a function of the static temperature, T, and the thermal conductivity, κ,

$$q_j = -\kappa\frac{\partial \widetilde{T}}{\partial x_j} \quad \text{where} \quad \kappa = \frac{\mu C_p}{Pr}\,. \tag{3}$$

The thermal conductivity is a function of the specific heat at constant pressure, Cp, of the Prandtl number, Pr, which is equal to 0.72 for air, and of the dynamic viscosity, μ, that can be calculated using the Sutherland law,

$$\mu\left(\widetilde{T}\right) = \mu_\infty\left(\frac{\widetilde{T}}{\widetilde{T}_\infty}\right)^{\frac{3}{2}}\frac{\widetilde{T}_0 + S_1}{\widetilde{T} + S_1}\,, \tag{4}$$

in which μ_∞, $\widetilde{T}_\infty$, $\widetilde{T}_0$ and S_1 are reference values. Density, static pressure and static temperature are correlated by the equation of state given by

$$\overline{p} = \rho\left(C_p - C_v\right)\widetilde{T}\,, \tag{5}$$

where C_v is the specific heat at constant volume. The shear-stress tensor, τ_{ij}, is written according to the Stokes hypothesis as

$$\tau_{ij} = \mu \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3}\delta_{ij} \frac{\partial \tilde{u}_k}{\partial x_k} \right) \tag{6}$$

The large eddy simulation set of equations can be written in a more compact form as

$$\frac{\partial \mathbf{Q}}{\partial t} = -\mathbf{RHS}\,, \tag{7}$$

where $\mathbf{Q}$ stands for the convervative properties vector and **RHS** represents the right hand side of Eq. 1, given by

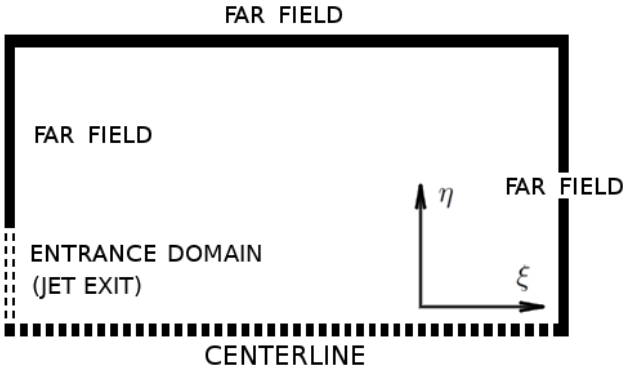
$$\mathbf{Q} = [\overline{\rho}\,, \overline{\rho}\widetilde{u_i}\,, \overline{e}]^T \quad \text{and} \quad RHS_i = \frac{\partial E_i}{\partial x_j} - \frac{\partial F_i}{\partial x_j}\,. \tag{8}$$

The components of inviscid and viscous flux vectors are respectively denoted by E_i and F_i, and written as

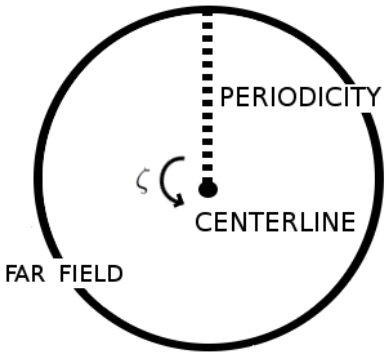
$$\mathbf{E} = \begin{bmatrix} \overline{\rho}\widetilde{u_j} \\ \overline{\rho}\widetilde{u_i}\widetilde{u_j} + \overline{p}\delta_{ij} \\ \left[(\overline{e} + \overline{p})\,\widetilde{u_j}\right] \end{bmatrix} \quad \text{and} \quad \mathbf{F} = \begin{bmatrix} 0 \\ \tau_{ij} \\ \tau_{ij}\widetilde{u_i} - q_j \end{bmatrix}. \tag{9}$$

Spatial derivatives are calculated in a structured finite difference context and the formulation is re-written for general curvilinear coordinate system [5]. The numerical flux is computed through a second-order central difference scheme with the explicit addition of anisotropic scalar artificial dissipation [29]. The time marching method is an explicit 5-stage Runge-Kutta scheme [30, 31].

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. Figures 1(a) and 1(b) indicate the boundary conditions used in the current work on lateral and frontal 2-D cuts of the domain. A supersonic flow is implemented at the entrance of the domain, also known by the aeroacoustic community as the jet exit of the nozzle, and Riemann invariants [32] are used to calculate the far field boundary conditions. The centerline of the computational domain is a singularity and it requires special treatment. Therefore, conserved properties are extrapolated from the adjacent longitudinal plane and they are averaged in the azimuthal direction in order to define the updated properties at the centerline of the jet. For the periodic plane, superposed points are used in the first and last points in the azimuthal direction in order to close the 3-D computational since it is a inner surface. The reader can find further details about the formulation in the work of *Junqueira-Junior et. al.* [11].



(a) Lateral view of boundary conditions.



(b) Frontal view of boundary conditions.

Figure 1: 2-D Lateral and frontal cuts of the computational domain indicating the position of boundary conditions.

4. Implementation Aspects

The current section presents details of the LES solver and discusses the high performance computing implementations introduced into the code. JAZzY is developed using the Fortran 90 standard and the parallel I/O concept. Recent improvements of the code include the implementation of asynchronous inter-partition communications and the development of a preprocessing tool that performs an optimized partitioning and creates multiple input grid files using a binary tree data structure. Each MPI rank reads its own input file when using the new preprocessing routine. The I/O modifications are also a first effort to bypass the original serial mesh allocation implemented in the code.

4.1. Preprocessing Mesh Partitioner

The LES solver presents a parallel I/O feature in which each MPI partition reads its correspondent portion of the mesh. Therefore, a 3-D grid partitioner is developed in order to provide partitioned mesh files to the solver. This preprocessing code can also generate grid files for simple geometric configurations. The mesh partitions are written using the CGNS standard [15, 16, 17, 18] which is built on the HDF5 library [13, 14]. This library is a general scientific format adaptable to virtually any scientific or engineering application. It provides tools to efficiently read and write data structured in a binary tree fashion. This data structure can handle many types of queries very efficiently [33, 34], such as time-dependent CFD solutions.

Figure 2 presents the flow chart of the mesh partitioner code. The preprocessing code can read a 2-D mesh file from a commercial grid generator or create a 2-D geometric configuration along with a grid point distribution using parameters provided by the user. Once the mesh is read by the preprocessing code, the 2-D domain is partitioned in the axial direction. After this

procedure, the partitioned mesh is extruded in the azimuthal direction, respecting the positioning of the MPI partitions. Each portion of the mesh is written using the CGNS standard.

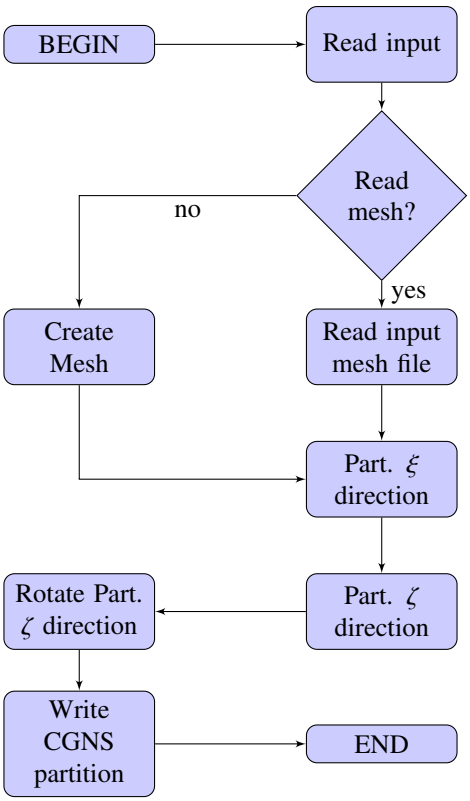
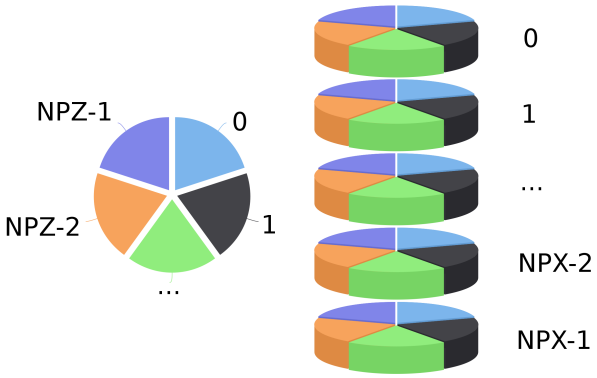


Figure 2: Preprocessing flow chart.

The partitioning of the domain is performed in a matrix fashion in order to create structured blocks which can be mapped and easily accessed through the use of message protocols. Figure 3(a) illustrates the segmentation of the domain into the axial and azimuthal directions while Fig. 3(b) presents the mapping of the domain. The index of each partition, indicated in Fig. 3(b), is based on a matrix index system in which the rows represent the position in the axial direction and the columns represent the position in the azimuthal direction. The partition index starts at zero to be consistent with the message passing interface standard. NPX and NPZ denote the number of partitions in the axial and azimuthal directions, respectively.

An optimized partitioning is necessary in order to have a well balanced distribution of tasks for all resources requested. The number of points within a given mesh can be a measure of computational costs for CFD applications using the finite difference spatial discretization. Therefore, the division of the mesh in the axial and azimuthal directions is performed towards a well balanced distribution of points. Firstly, the total number of grid points in one direction is divided by the number of domains in the same direction. The remaining points are spread among the partitions when the division is not exact. The same procedure is performed in the other directions. Figure 4 illustrates the balancing procedure performed in each direction during the



(a) 2-D partitioning in the axial and azimuthal directions.

0	NPZ	2*NPZ		
1	NPZ+1	(2*NPZ)+1		
(...)	(...)	(...)		
NPZ-2	NPZ+NPZ-2	2*NPZ+NPZ-2	j*(NPZ)+i	
NPZ-1	NPZ+NPZ-1	2*NPZ+NPZ-1	(...)	(NPX*NPZ)-1

(b) Mapping of the 2-D partitioning.

Figure 3: 2-D partitioning and mapping.

partitioning of the computational grid, in which n stands for the integer part of the division and m represents the number of points in the unbalanced partition.

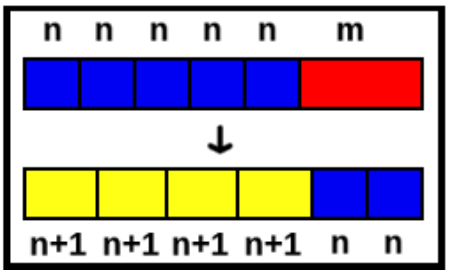


Figure 4: Balancing procedure performed during the patitioning of the mesh.

4.2. *Large Eddy Simulation Code*

JAZzY is the LES solver used in the current work. A brief overview of the code is presented as a flow chart illustrated in Fig. 5. Initially, every MPI partition reads the same ASCII file which provides input data such as flow configurations and simulation settings. In the sequence, each MPI partition reads its corresponding CGNS mesh file. The Jacobian and the metric terms are calculated after the I/O procedure. Then, each MPI rank sets the initial conditions defined in the input data. The

Runge-Kutta time integration is the first routine to be called into the interaction loop.

At the beginning of the time marching scheme, for each substep, asynchronous communications of the conservative property vector are performed, in both, azimuthal and axial directions, followed by an update of boundary conditions and dynamic viscosity coefficient. Viscous terms are communicated before the computation of artificial dissipation operators and viscous flux vectors. MPI waiting functions are carefully added along the code to avoid out-dated information and to enforce the preservation of the numerical method accuracy. Finally, when the requested number of time steps is achieved, each MPI partition appends the solution to the output CGNS file.

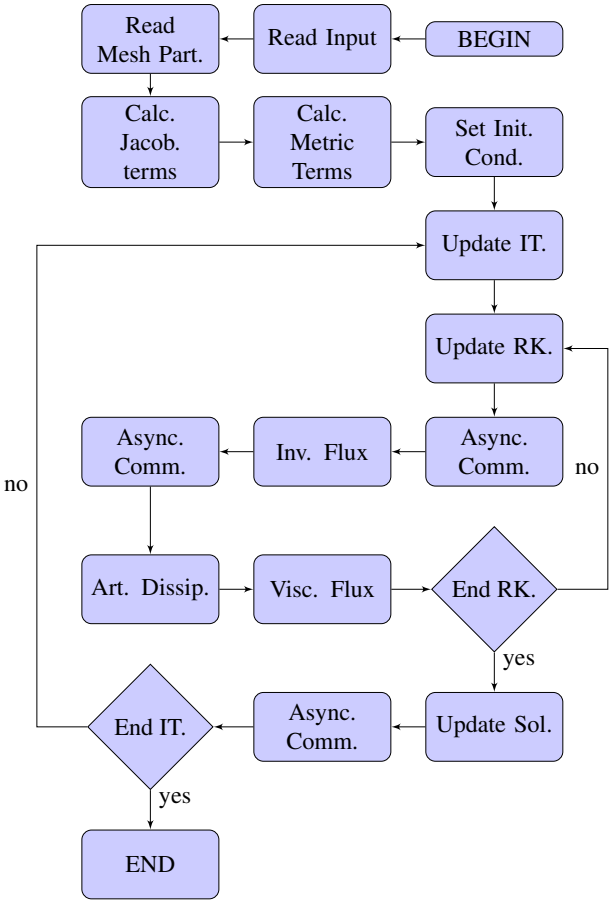


Figure 5: JAZzY solver flow chart.

4.3. Inter-Partition Data Exchange

In the present work, data exchanges are performed using ghost points which are added to the boundaries of local mesh partitions at the main flow direction and at the azimuthal direction in order to carry information of the neighboring points. The artificial dissipation scheme implemented in the code [30] uses a five-point stencil which demands information of two neighbors, in each side, of a given mesh point. Hence, a two-layer ghost point fringe is created at the beginning and at the end of each partition. Figure 6 illustrates the ghost points of a partition domain. The yellow and black layers represent the axial and azimuthal ghost points, respectively, while the green region is the partition mesh.

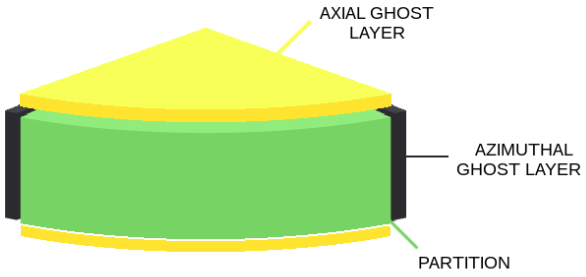
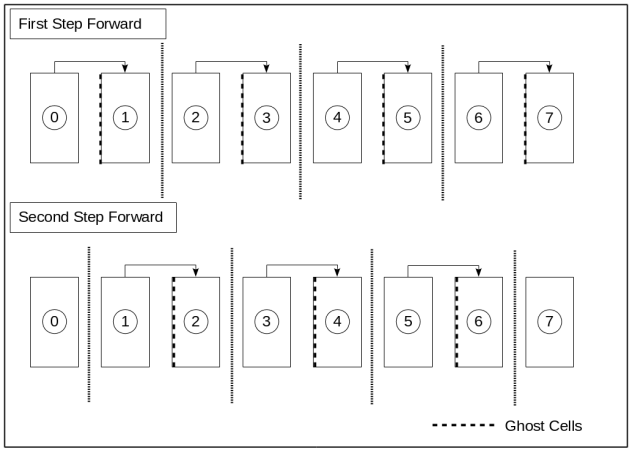


Figure 6: Ghost points creation procedure. The green volume indicates one given partition of the domain, while the yellow and black layers represent the axial and azimuthal ghost points layers, respectively.

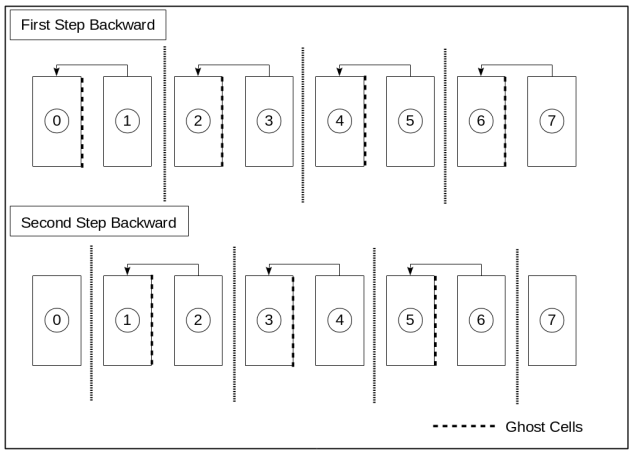
The solver was originally developed using a four-step blocking communication for both axial and azimuthal directions. Figures 7(a) and 7(b) demonstrate the data exchange approach for a given direction. Initially, the communication is performed in the forward direction where even partitions send information of their two last local layers to the ghost points at the left of odd partitions. If the last partition is even, it does not share information in this step. In the sequence, odd partitions send information of their two last local layers to the ghost points at the left of even partitions. If the last partition is odd, it does not share information in this step. The third and fourth steps are backward communications which are adjusted to work in the same fashion as the forward communication does.

The four-step blocking communications are replaced by asynchronous MPI protocols in order to improve the performance of the solver. Waiting functions are carefully added along the Runge-Kutta calculations in order to avoid the presence of outdated information into the ghost cells and to preserve the accuracy of the time integration method.

The meshes used in the current research have a singularity at the centerline. It is necessary to correctly treat this region for the sake of data consistency. Therefore, properties are extrapolated to the singularity in the radial direction and, in the sequence, the master partition collects all data from the partitions that share the same singularity point and allocates such information into one single vector. After the allocation, the properties are averaged in a sequential fashion and the result is scattered to the neighbors in the azimuthal direction. Such procedure does not use collective communications and it is performed in a blocking communication fashion in order to preserve the commutative property during the averaging. This blocking communication is very important in order to achieve the binary reproducibility of the computational tool [35]. This approach assures that parallel computations treat the centerline singularity in the exactly same fashion sequential calculations do. The use of such communication is motivated by the work of Arteage *et. al.* [36]. Moreover, such approach can help de-



(a) Forward blocking communication.



(b) Backward blocking communication.

Figure 7: Forward and backward communication approaches originally implemented in the code in order to exchange information between neighboring partitions.

velopers verify whether new parallel features implemented into the code present the same behavior when running sequentially. The singularity treatment is the only blocking data exchange present into the LES solver after the last upgrades.

5. Computational Performance Study

Large eddy simulations require a significant amount of computational resources in order to produce high-quality results. Therefore, it is very important to evaluate the parallel tool to be sure that it is capable of using parallel resources efficiently. Data exchange between partitions is not free and it can affect the parallel computational performance.

5.1. Scalability Setup

Isothermal perfectly expanded jet flow simulations are performed using different grid sizes and different partition configurations. The Reynolds number of the jet is 1.5744×10^6 for the present simulations and a flat-hat profile with Mach number of 1.4 is imposed at the jet exit of the nozzle. Isothermal and zero-velocity conditions, along with a zero-pressure-gradient state, consistent with the freestream condition, are used as initial conditions for the simulations. The JAZzY solver has already been validated and presented good results using such flow configuration [11]. The reader can find more details about this flow configuration in the *Compressible Jet Flow Simulation* section. In the present work, each simulation performs 1000 iterations or 24 hours of computation, whatever is shorter in terms of computational time. An average of the CPU time per iteration through the simulations is measured in order to evaluate the weak and strong scalability of the solver using the Santos Dumont supercomputer.

One geometry is created for the computational evaluation, where the 2-D surface of this computational domain, as presented in Fig. 1(a), is 30 dimensionless units in length and 10 dimensionless units in height. The diameter of the jet exit is the length reference unit. In the present work, 13 grids of different sizes are created, starting with the coarsest mesh with 370,000 points up to the finest mesh, with approximately 1 billion grid points, as indicated in Tab. 2. The mesh size growth of the first 12 grids follows a geometric progression with ratio 2.

Table 2: Mesh configurations used for the scalability study.

Mesh	Nb. Pt. ξ	Nb. Pt. η	Nb. Pt. ζ	Nb. Pt.
1	32	32	361	$\approx 370k$
2	64	32	361	$\approx 740k$
3	64	64	361	$\approx 1.5M$
4	128	64	361	$\approx 3.0M$
5	128	128	361	$\approx 6.0M$
6	256	128	361	$\approx 11.8M$
7	256	256	361	$\approx 23.7M$
8	512	256	361	$\approx 47.3M$
9	512	512	361	$\approx 94.6M$
10	1024	512	361	$\approx 190M$
11	1024	1024	361	$\approx 380M$
12	2048	1024	361	$\approx 760M$
13	1700	1700	361	$\approx 1.0B$

Simulations performed in the present study are run using up to 3072 computational cores. Different partitioning configurations are evaluated for a given number of processors. Table 3 presents the number of partitions in the azimuthal direction, NPZ, for a given number of computational cores. The partitioning configurations which provided the fastest calculation are used to evaluate the scalability of the solver. Table 4 presents the number of ghost points as a percentage of the number of grid points, from the optimal partitioning configuration, for all mesh configurations and computational cores used in the strong scaling study. One can understand the ghost point per total grid point ratio as the communication cost for a parallel calculation using MPI. In the present work, such cost is, for most of the partitioning configurations, less than 1% and approximately 7% in the worst case scenario.

Table 3: Grid partitioning configurations for different number of cores.

Nb. Cores	**NPZ**					
2	1	2				
4	1	2	4			
8	1	2	4	8		
16	1	2	4	8	16	
32	1	2	4	8	16	32
64	2	4	8	16	32	
128	2	4	8	16	32	
256	4	8	16	32		
512	4	8	16	32		
1024	8	16	32			
2048	8	16	32			
3072	24	48	96			

5.2. *Strong Scaling*

The speedup is one of the most common figures of metric for the performance evaluation of parallel algorithms and architectures [37] and it is used in the present work in order to measure the strong scaling of the solver and compare it with the ideal case. Different approaches are used by the scientific community in order to calculate the speedup [38, 39]. In the present work the speedup, $S\,p(\mathbf{m}, N)$, is given by

$$S_{\,p}(\mathbf{m}, N) = \frac{T(\mathbf{m}, s)}{T(\mathbf{m}, N)}\,. \tag{10}$$

In this expression, T stands for the time spent by $\mathbf{m}$-th mesh to perform one thousand time steps, 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 [40] as

$$\eta^{st}(\mathbf{m}, N) = \frac{S\,p(\mathbf{m}, N)}{N}\,. \tag{11}$$

Ideally, the sequential computation is the starting point for a strong scalability study. However, quite often, the computational problem is too big to fit in one single cluster node due to hardware limitations of the computer node. Hence, it is necessary to shift the starting point to a minimum number of processors in which the code can be run. For such cases, where $N = s$, speedup and strong scaling efficiency are assumed as

$$S_{\,p}(\mathbf{m}, s) = s \quad \text{and} \quad \eta^{st}(\mathbf{m}, s) = 1\,. \tag{12}$$

Table 5 presents the minimum and maximum number of computational cores used for each mesh configuration in the strong scaling study.

5.2.1. *Evaluation of Code Improvements*

An initial evaluation of the improvements brought to the code is performed in a smaller supercomputer before studying the scalability of the new version of the solver in the Santos Dumont supercomputer. Optimal workload distribution and non-blocking communications are the main modifications delivered to the LES solver. A strong scalability comparison is fulfilled using the Euler supercomputer which is included into a national project known as CEPID-CeMEAI [41]. The cluster is an Hewlett Packard Enterprise (HPE) machine. It presents 104 computational nodes and each one has two deca-core 2.8 GHz *Intel Xeon*® E52680v2 processors and 128 Gb DDR3 1866MHz random access memory. The entire cluster has 2080 computational cores available for the project members.

Simulations are run using mesh 8 configuration, which has $\approx$ 50 million grid points, as indicated in Tab. 2. The study is performed allocating up to 400 computational cores in parallel. Figure 8 presents a comparison between the strong scalability from both versions of the code. One can observe that the strong efficiency curve is shifted up approximately by 20% with the improvements implemented in the code for the grid configuration evaluated. The results enhance the importance of the latest upgrades brought to the JAZzY solver.

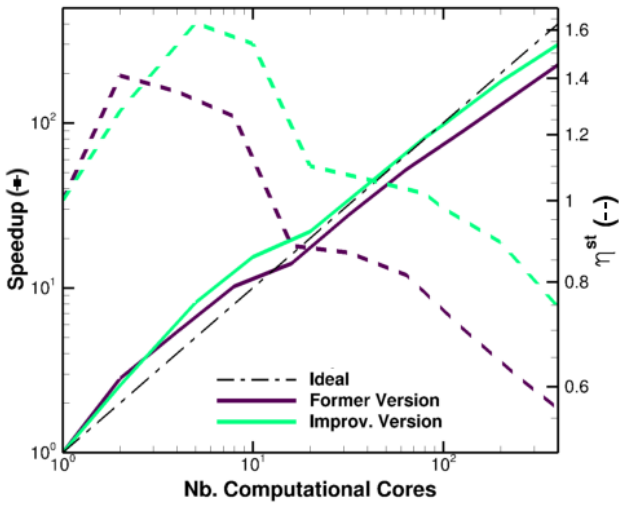


Figure 8: Comparison of speedup (–) and strong scaling efficiency (- -) curves from both versions of the JAZzY solver over approximately 50 million grid points, *i.e.*, mesh 8.

5.2.2. *Strong Scalability on Santos Dumont Supercomputer*

After the evaluation of recent modifications, a more detailed scalability study is done using the Teraflop branch from the Santos Dumont supercomputer. The LES solver, used in the present study, can run up to $\approx$ 50 Million grid points, *i.e.*, mesh 8 configuration, in one single node of the Santos Dumont machine, which has 64GB of RAM. The standard maximum number of computational cores allowed to be used for scalability tests in the Brazilian computer is 3072. More than 400 simulations are run in order to perform the scalability study, considering all mesh configurations and different partitioning arrangements.

Figures 9, 10 and 11 present the speedup, in solid line, and the strong efficiency, in dashed line, for the 13 mesh configurations. One can notice that the speedup and strong efficiency of all numerical test cases present similar trends. The speedup

Table 4: Number of ghost points as a percentage of the total number of grid points for every grid and amount of computational resources used in the strong scaling study.

Mesh	Nb. Cores 2	4	8	16	32	64	128	256	512	1024	2048	3072
1	0.81%	0.52%	0.66%	0.93%	1.46%	1.82%	2.53%	3.96%	6.80%			
2	0.26%	0.33%	0.47%	0.74%	0.93%	1.46%	1.82%	2.53%	3.96%			
3	0.21%	0.17%	0.23%	0.37%	0.64%	0.73%	0.91%	1.27%	1.98%			
4	0.08%	0.13%	0.17%	0.23%	0.60%	0.64%	0.73%	0.91%	1.27%			
5	0.04%	0.07%	0.08%	0.16%	0.16%	0.32%	0.33%	1.58%	1.60%	1.63%		
6	0.03%	0.04%	0.06%	0.08%	0.13%	0.30%	0.23%	0.45%	1.58%	1.60%		
7	0.02%	0.02%	0.04%	0.08%	0.06%	0.08%	0.12%	0.18%	0.23%	0.80%		
8	0.01%	0.02%	0.03%	0.04%	0.05%	0.06%	0.08%	0.13%	0.23%	0.42%		
9					0.02%	0.03%	0.05%	0.11%	0.09%	0.21%	0.40%	0.16%
10						0.02%	0.04%	0.05%	0.04%	0.09%	0.21%	0.13%
11							0.02%	0.02%	0.04%	0.05%	0.06%	0.07%
12								0.02%	0.01%	0.04%	0.05%	0.06%
13								0.01%	0.02%	0.02%	0.03%	0.03%

Table 5: Starting point, s, and maximum number of computational cores, N_{max}, used by each grid configuration.

Mesh	s	N_{max}	Mesh	s	N_{max}
1	1	512	8	1	1024
2	1	512	9	32	3072
3	1	512	10	64	3072
4	1	512	11	128	3072
5	1	1024	12	256	3072
6	1	1024	13	256	3072
7	1	1024			

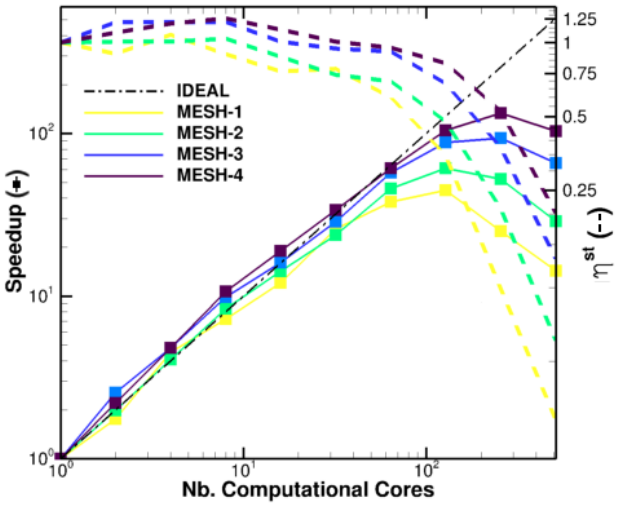


Figure 9: Speedup (–) and strong efficiency (- -) of meshes 1, 2, 3 and 4.

proportionally increases with the number of processors until it reaches a saturation point, which indicates that MPI data exchange is becoming as time consuming as the calculations themselves. This fact is reinforced by the value of the strong scalability efficiency of approximately 50% at the vicinity of the speedup peak, for all meshes evaluated.

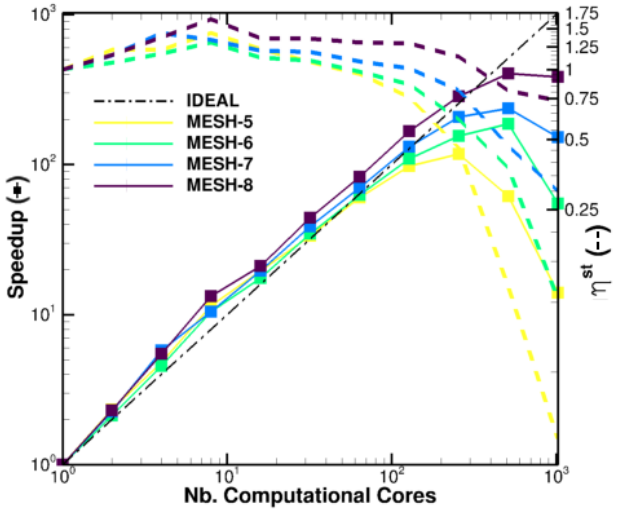


Figure 10: Speedup (–) and strong efficiency (- -) of meshes 5, 6, 7 and 8.

The two smallest problems evaluated in the present paper present similar strong scaling behavior. In both cases, the scalability curves follow the ideal case for $N \leq 8$ and they reach a maximum speedup of approximately 60 using 128 computational cores. The results with Meshes 3 and 4 indicate a higher value of speedup peak, which is approximately 100 for $N = 256$, when compared to the two smallest problems. Moreover, they present super-scalability when running with less than 16 partitions in parallel and $\eta^{st} \approx 1$ for $32 < N < 64$.

A similar shape for the speedup curves, as observed in Fig. 9, can also be seen in Fig. 10. However, the speedup saturation point is shifted upwards with the increase in the size of the computational problem. Mesh 5 configuration presents a maximum speedup of ≈ 100 for $N = 256$, while Mesh 6 test case has a speedup peak of ≈ 200 when running on 512 cores in

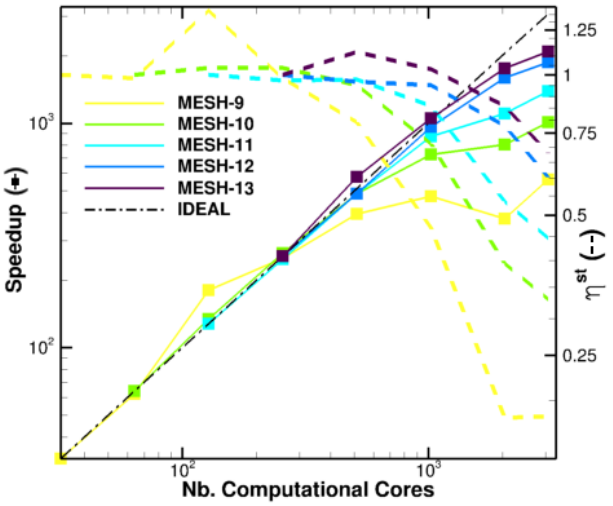


Figure 11: Speedup (–) and strong efficiency (- -) of meshes 9, 10, 11, 12 and 13.

parallel. Both cases have super-linear scaling for $N \leq 32$ and strong efficiency of 100% when running on 64 MPI partitions. Meshes 7 and 8 present the speedup apex of ≈ 250 and ≈ 500, respectively, for $N = 512$. The former case indicates $\eta^{st} \geq 1$ for $N < 256$, while the latter presents super-scaling for $N \leq 512$.

The last five cases, presented in Fig. 11, which have more than 90 million grid points, indicate better strong scaling when compared to the previous test cases. For these five cases, the speedup scales proportionally to the number of available resources. Meshes 9, 10 and 11 present $\eta^{st} \geq 1$ for $N \leq 256$, although the efficiency curves are reasonably flat and around $\eta^{st} \approx 1$ for meshes 10 and 11 all the way up to $N \leq 1024$. The largest grid size analyzed here, *i.e.*, mesh 13, presents $\eta^{st} \approx 1$ for calculations performed using up to 2048 partitions and high strong efficiency, $\eta^{st} \approx 70\%$, when using 3072 computational cores in parallel.

The presence of super-linear scalability, as observed in Figs. 9, 10 and 11, can be explained by fact that cache memory no longer becomes a bottleneck with the increase of the number of computational resources. Since more cache memory is available, when the number of processors is increased, it can be used more effectively by the workload in each process when compared to the use of cache memory of starting point calculation of the strong scalability study [42, 43]. The cache memory bottleneck could be related to a non-optimal memory access design of the code. Furthermore, it could be correlated to the deterioration of performance when increasing the number of processors used in a calculation. The code would presumably not deliver optimal efficiency when using hundreds of thousands of cores in parallel. The developers are nevertheless investigating the cause of such issues. Implementing cache blocking [44, 45, 46] and vectorization [47, 48] techniques could overcome the difficulty simultaneously with an eventual further improvement of the solver performance in parallel.

It is important to remark that the authors are interested on simulations of supersonic jet flows. Calculations of this flow configuration, using the LES formulation, require meshes with more than 100 million points. The present study indicates that the JAZzY solver has a good performance for problems of this size and it is capable of using the Santos Dumont supercomputer efficiently. Moreover, the strong scaling evaluation can be used as a guideline for the selection of the best partitioning configuration and/or the number of processors to be used in the Brazilian cluster, for a given mesh size required for the particular study.

5.3. Weak Scalability on Santos Dumont Supercomputer

Weak scalability can be interpreted as the ability of conserving the computing time for a fixed workload. In the ideal case, a parallel code should preserve constant time-to-solution when the dimension of a problem increases at the same rate as the number of computational resources. The weak scaling efficiency for a given workload, w, can be written mathematically as

$$\eta^{wk}(w, N) = \frac{T(w, s)}{T(w, N)} \,. \tag{13}$$

The weak scaling of the parallel solver is evaluated using 5 different workloads, which are represented in the present paper by the number of grid points divided by the number computational cores. Table 6 presents the mesh and the number of processing units used at the starting point followed by the maximum number of computing units and the workload for each weak scalability study. The computational grids, presented in

Table 6: Description of workloads and starting points used for the weak scalability study.

Case	1st Mesh	s	N_{max}	Workload $\left[\frac{\text{Nb. Pt.}}{\text{Core}}\right]$
A	Mesh 1	2	2048	$\approx 185k$
B	Mesh 1	1	2048	$\approx 370k$
C	Mesh 2	1	2048	$\approx 740k$
D	Mesh 3	1	1024	$\approx 1.5M$
E	Mesh 4	1	512	$\approx 3.0M$

Tab. 2, are created using a total number of mesh points which is proportional to the power of two in order to be able to keep a constant workload when the computational resources of a given test case is doubled. However, due to limitations of the parallel tool, Mesh 13 configuration is an exception to such trend and presents $\approx 1.0B$ points, which is ≈ 1.45 the size of Mesh 12 test case that has $\approx 760M$ points. Therefore, for this special grid configuration, a correction is used on the calculation of η^{wk} in order to properly compare it with data collected from other mesh configurations. It is also important to remark here that, the first case is the only one to use two computational cores as a starting point, s. The other weak scaling studies are performed using a sequential computation as a starting point.

Figure 12 presents the weak scalability of the five different workloads evaluated in the current paper. In the worst case sce-

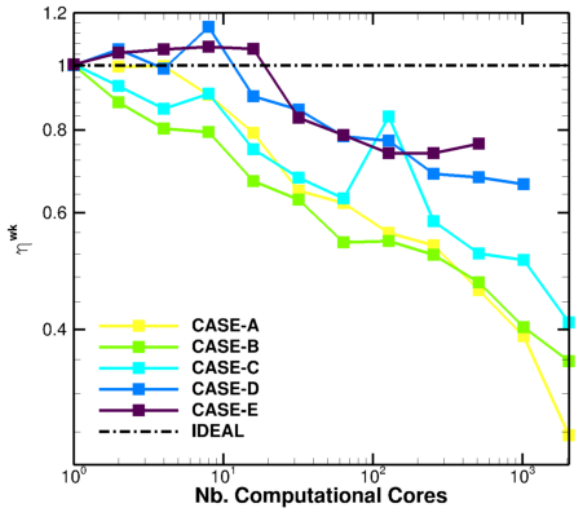


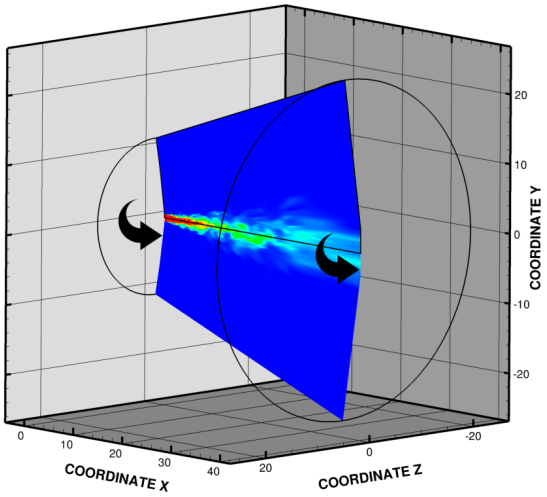
Figure 12: Weak scalability of the JAZzY solver running on the Santos Dumont supercomputer.

nario, the parallel code can still present a good weak scalability with $\eta^{wk} \approx 0.7$ for the two largest workloads evaluated in present work, 1.5M and 3.0M grid points per core. Nonetheless, one can notice the weak scaling efficiency decay when increasing the number of processors used on the calculations. Moreover, the super-linear speedup behavior, previously observed on the strong scaling study, matches with weak scalability peaks. One example is the sudden rise of η^{wk} present on the Case C scaling illustrated in Fig. 12 for 128 computational cores. Such event matches the strong super-scalability behavior of Mesh 9 test case for 128 processing units indicated in Fig. 11.

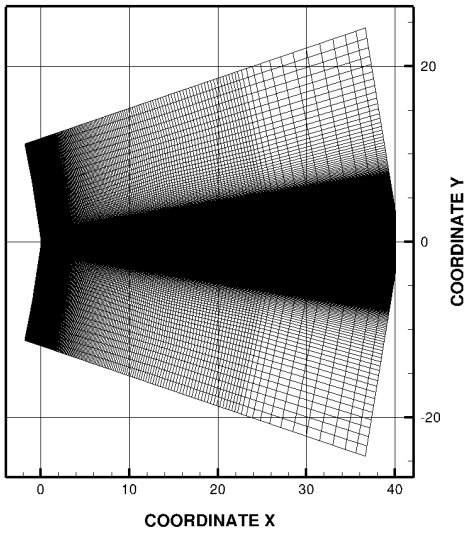
6. Compressible Jet Flow Simulation

This section presents results achieved from the simulation of a supersonic jet flow configuration. This calculation is 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 [49] and to experimental data [50]. The reader can find more details of this particular simulation in the work of *Junqueira-Junior et. al.* [11].

A geometry is created using a divergent shape whose axis length is 40 times the diameter of the jet exit, D. The minimum and maximum heights of the domain are $\approx 16D$ and $25D$, respectively. Figure 13 illustrates a 2-D cut of the geometry and the grid point distribution used on the validation of the solver. The mesh created to validate the parallel solver is composed of 343 points in the axial direction, 398 points in the radial direction and 360 points in the azimuthal direction. This yields a grid with approximately 50 million points. The distance between mesh points increases towards the outer region of the domain. This procedure forces the dissipation of properties far from the jet exit of the nozzle in order to avoid reflection of waves into the domain. The grid coarsening can be understood as an implicit damping which can smooth out properties far from the entrance domain, in the region where the mesh is no longer refined. The calculation is performed using 500 computational cores.



(a) 2-D cut of the geometry colored by velocity magnitude contours.



(b) 2-D cut of the domain superimposed by grid points distribution.

Figure 13: Illustration of geometry and mesh used into the validation of the LES solver.

An isothermal perfectly expanded jet flow is studied for the present validation. The Mach number at the exit of the nozzle is $M = 1.4$. The pressure ratio, $PR = P_j/P_\infty$, and the temperature ratio, $TR = T_j/T_\infty$, between the entrance of the domain 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 exit of the nozzle and the ∞ subscript stands for properties at the farfield region. The Reynolds number of the jet is $Re = 1.57 \times 10^6$, based on the diameter of the domain entrance, D.

The time increment, Δt, used for the validation study is 1×10^{-4} dimensionless time units.

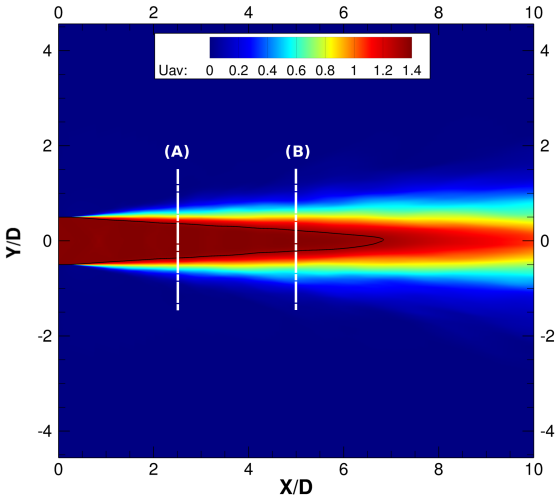
The boundary conditions previously presented in the *Large Eddy Simulation Formulation* section are applied in the current simulation. Initial conditions of the computation are set as isothermal and zero-pressure-gradient states, consistent with to the freestream condition, along with stagnated velocity, *i.e.*, zero velocity. The simulation runs a predetermined period of time until the statistically steady flow condition is achieved. This first pre-calculation is important in order to assure that the jet is fully developed and turbulent.

In the present work, the pre-calculation is run for 14 flow through time (FTT) units before reaching the statistically steady flow condition. One FTT unit represents the necessary amount of time for a particle to cross all the domain, in the main flow direction, considering the inlet velocity at the jet exit of the nozzle. After the statistically stationary state is achieved, the simulations are restarted and run for 3.30 FTT more in which data of the flow are extracted and recorded in a frequency of 50Hz. The data sampling can be as well represented by a Strouhal number range of $0.01 \lesssim S_t \lesssim 36$. The Strouhal number, S_t, is calculated based on the frequency of extraction, Mach number at the exit of the nozzle, and the jet diameter.

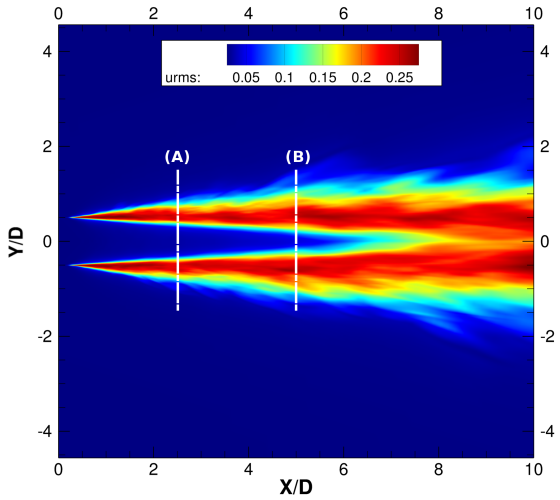
Figure 14 indicates the positioning of the two surfaces, (A) and (B), where data are extracted and averaged through time. Cuts (A) and (B) are radial profiles at $2.5D$ and $5.0D$ units downstream of the domain entrance. Flow quantities are also averaged in the azimuthal direction when the radial profiles are calculated. Moreover, Figs. 14(a) and 14(b) present 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. 14(a) 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. 15 and 16, respectively. The solid line stands for results achieved using the JAZzY code while square and triangular symbols represent numerical [49] and experimental [50] data, respectively. The averaged profiles obtained in the present work correlate well with the reference data at the two positions compared here. Nevertheless, the results achieved for both the JAZzY solver and by the numerical reference [49] present difficulties to correctly predict the peaks of u^*_{RMS}/U_j near $r/D = 0.5D$ at $2.5D$ when compared with experimental data [50], as indicated in Fig. 16. 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. Therefore, efficient massive parallel computing is mandatory in order to achieve high-quality results.

Figures 17 and 18 present a lateral view of an instantaneous visualization of the pressure contours, in gray scale, superimposed by 3-D velocity magnitude contours and vorticity magnitude contours respectively, in color, calculated by the LES tool



(a) Time averaged axial component of velocity, $\langle U \rangle$.



(b) RMS values of the fluctuating part of velocity axial component, u^*_{RMS}.

Figure 14: 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.

discussed in the present paper. A detailed visualization of the entrance domain is shown in Fig. 18(b). 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 present work is concerned with an evaluation of the performance of a computational fluid dynamics tool for

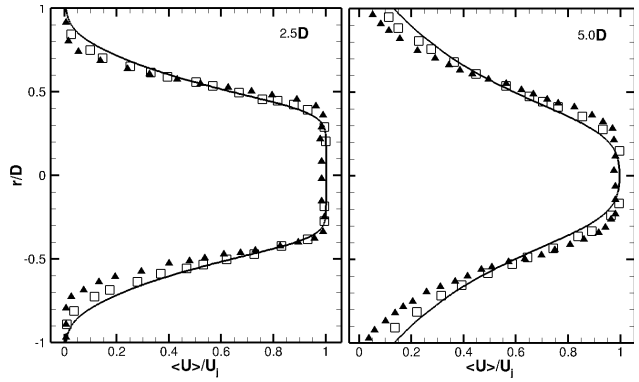


Figure 15: Profiles of the averaged axial component of velocity, $\langle U \rangle$, at $2.5D$ and $5.0D$ from the entrance: (–) JAZzY results; (□) numerical data; (▲) experimental data.

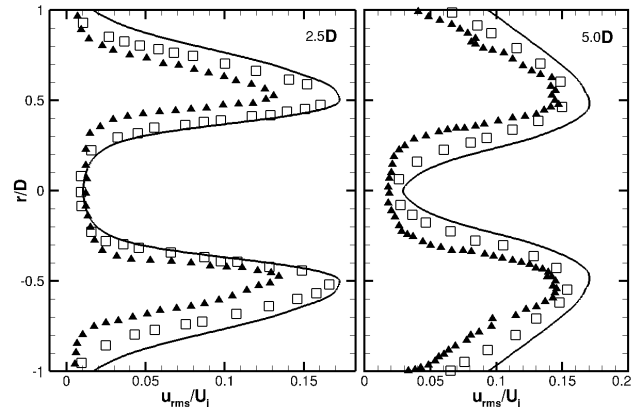


Figure 16: Profiles of the RMS of the fluctuation part of axial component of velocity, u^*_{RMS}, at $2.5D$ and $5.0D$ from the entrance: (–) JAZzY results; (□) numerical data; (▲) experimental data.

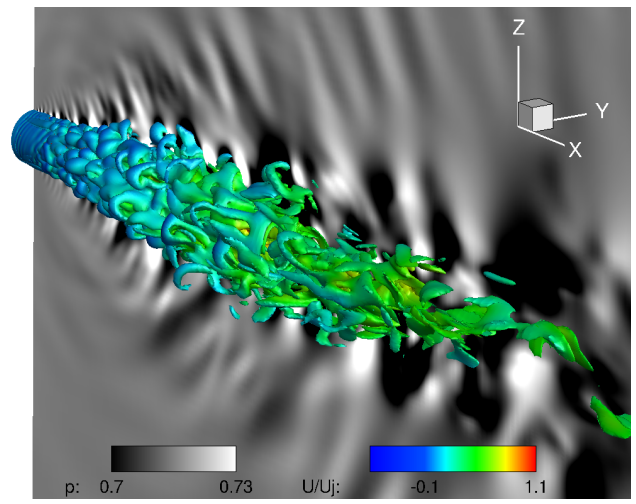
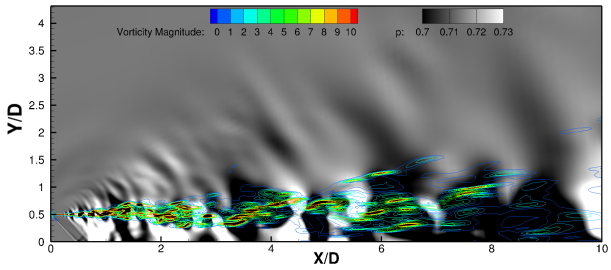
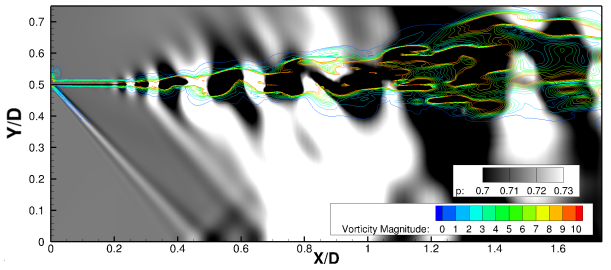


Figure 17: Instantaneous lateral view of pressure contours, in gray scale, superimposed by 3-D velocity magnitude contours, in color.

aerospace applications when using a national supercomputer from the Brazilian National Laboratory for Scientific Computing (LNCC). The cluster is named Santos Dumont and it is in the 193rd position of the top 500 list of November 2019. The



(a) Lateral view of pressure and magniture of vorticity.



(b) Detailed view of pressure and magnitude of vorticity at the jet exit of the nozzle.

Figure 18: Lateral and detailed view of pressure contours, in gray scale, superimposed by vorticity magnitude contours, in color.

numerical solver was developed by the authors to study supersonic jet flow configurations using a large eddy formulation. Results obtained with the code are used for aeroacoustic design applications. The simulations of such flow configurations are expensive and require continuous improvement of the numerical tool regarding efficient parallel computing. Therefore, weak and strong scalability studies of the solver are performed on the Santos Dumont supercomputer in order to evaluate if the numerical tool is capable of efficiently using thousands of processors in parallel. It should be noted that study is performed in the Teraflop partition of the Santos Dumont supercomputer which has 18,144 computational cores, 396 Nvidia K40 graphical processing units and 108 Xeon Phi 7120 Intel accelerators. Moreover, the system section has a LINPACK performance of 456.8 TFlop/s and a theoretical peak performance of 657.5 TFlop/s.

Spatial discretization in the solver uses a second-order centered finite difference approach. Time integration is performed using a five-stage explicit Runge-Kutta scheme. The code is implemented using Fortran 90 standards coupled with message passing interface (MPI) for inter-partition communications. The jet flow-like geometry and flow condition were defined for the scalability study, and 13 different grid refinement levels were constructed. Furthermore, different partitioning configurations were used in order to evaluate the parallel code under different workloads. Grid sizes were varied from $370,000$ to approximately 1.0 billion grid points. Calculations were performed for 1000 time steps or 24 wall-clock hours of computation, whichever was reached first, using up to 3072 cores in parallel. The CPU time per iteration was averaged, when the simulation was finished, in order to calculate the speedup and scaling efficiency. A preliminary strong scaling

study is performed using a smaller computer to evaluate the effects from the recent implementations added to the LES solver using up to 400 computational cores in parallel. Hence, more than 400 simulations were performed for the scalability study of the parallel solver in the Santos Dumont supercomputer.

The preliminary strong scalability study presents an increase of 20% on the efficiency curve when using a configuration of 50-million grid points along with up to 400 computational cores in parallel. The strong and weak scaling studies performed in the Santos Dumont computer indicate that the parallel tool has a good strong scalability and it is able to speedup the time-to-solution when running on more than 3000 processing units with a strong scalability efficiency of approximately 70%. Super-linear strong scaling is also observed in the tests performed. Moreover, the largest mesh configuration addressed in the present effort has achieved a strong scalability which follows the ideal case very close, even when running on 2048 computational cores.

Five different workloads are considered in the present work in order to study the weak scaling of the solver. Tests are performed starting with 165×10^3 grid points per computing unit and moving up to 3.0×10^6 grid points per core. The load is doubled for each weak scalability test to the next one. The first two test cases present a significant decay on the weak scaling efficiency when increasing the number of processors. The performance for a fixed number of processors is improved for higher workloads. In the worst scenario evaluated in the present work, the test cases using 1.5 and 3.0 million points per core present a weak scalability efficiency of $\approx 70\%$. It is also important to remark that super-linear speedup events indicated in the strong scalability studies are related to the presence of efficiency peaks in the weak scaling curves.

The results, obtained for the physically relevant test case, indicate that it is possible to achieve good results for supersonic jet flows using the present solver. The simulations performed to validate the numerical code are in good agreement with experimental and numerical references in the regions where the grid presents high resolution. Additionally, the present work indicates that the parallel implementation of the code is capable of handling high spatial resolution properly. Furthermore, the scalability study performed in the current paper can serve as a guide for future simulations using the same numerical tool in the Santos Dumont supercomputer. Nevertheless, the authors recognize the existence of improvement possibilities and will continue the development of the solver regarding the optimization of cache memory access and the implementation of vectorization capabilities.

Acknowledgments

The authors gratefully acknowledge the partial support for this research provided by Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, under the Research Grants # 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 Grants # 2013/07375-0 and 2013/21535-0. The authors further acknowledge the National Laboratory for Scientific Computing, LNCC/MCTIC, for providing high-performance computing resources through the Santos Dumont supercomputer, which have been indispensable in order to obtain the research results reported in this paper.

References

[1] P. M. Nelson, Transportation Noise Reference Book, Butterworth-Heinemann, Oxford, UK, 1987.

[2] M. Darecki, C. Edelstenne, T. Enders, E. Fernandez, P. Hartman, J. P. Herteman, M. Kerkloh, I. King, P. Ky, M. Mathieu, G. Orsi, G. Schotman, C. Smith, J. D. Wörner, Flightpath 2050: Europes vision for aviation, Tech. rep., European Commission (2011).

[3] P. Argüelles, M. Bischoff, P. Busquin, B. A. C. Droste, S. Evans, W. Kröll, J. L. Lagardere, A. Lina, J. Lumsden, D. Ranque, S. Rasmussen, P. Reutingler, S. R. Robins, H. Terho, A. Wittlöv, European aeronautics: A vision for 2020, Tech. rep., European Commission (2001).

[4] C. Wagner, T. Hüttl, P. Sagaut, Large-Eddy Simulation for Acoustics, Cambridge University Press, Cambridge, UK, 2007.

[5] C. A. Junqueira-Junior, Development of a parallel solver for large eddy simulation of supersonic jet flow, Ph.D. thesis, Instituto Tecnoógico de Aeronáutica, São José dos Campos, SP, Brazil (2016).

[6] E. Garnier, N. Adams, P. Sagaut, Large Eddy Simulation for Compressible Flows, Springer, 2009.

[7] D. Bodony, S. K. Lele, On using large-eddy simulation for the prediction of noise from cold and heated turbulent jets, Physics of Fluids 17 (8).

[8] W. Wolf, S. K. Lele, Airfoil aeroacoustics: LES and acoustic analogy predictions, Ph.D. thesis, Stanford, Stanford, CA, USA (2011).

[9] S. C. Lo, K. M. Aikens, G. A. Blaisdell, A. S. Lyrintzis, Numerical investigation of 3-D supersonic jet flows using large-eddy simulation, International Journal of Aeroacoustics 11 (7) (2012) 783–812.

[10] W. R. Wolf, J. L. F. Azevedo, S. K. Lele, Convective effects and the role of quadrupole sources for aerofoil aeroacoustics, Journal of Fluid Mechanics 708 (2012) 502–538.

[11] C. Junqueira-Junior, S. Yamouni, J. L. F. Azevedo, W. R. Wolf, Influence of different subgrid-scale models in low-order les of supersonic jet flows, Journal of the Brazilian Society of Mechanical Sciences and Engineering 40 (258) (2018) 1–29.

[12] J. J. Dongarra, S. W. Otto, M. Snir, D. Walker, An Introduction to the MPI Standard, Tech. rep., Knoxville, TN, USA (1995).

[13] M. Folk, G. Heber, Q. Koziol, E. Pourmal, D. Robinson, An overview of the HDF5 technology suite and its applications, in: Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, ACM, 2011, pp. 36–47.

[14] M. Folk, A. Cheng, K. Yates, HDF5: A file format and I/O library for high performance computing applications, in: Proceedings of Supercomputing, Vol. 99, 1999, pp. 5–33.

[15] C. L. Rumsey, B. Wedan, T. Hauser, M. Poinot, Recent updates to the CFD general notation system (CGNS), in: *AIAA Paper No. 2012-1264, Proceedings of 50th AIAA Aerospace Sciences Meeting, Nashville, TN, USA, 2012, p. 16.*

[16] S. M. Legensky, D. E. Edwards, R. H. Bush, D. Poirier, CFD general notation system (CGNS) - status and future directions, in: AIAA Paper No. 2002-0752*, Proceedings of 40th AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV, 2002.*

[17] D. M. A. Poirier, R. H. Bush, R. R. Cosner, C. L. Rumsey, D. R. McCarthy, Advances in the CGNS database standard for aerodynamics and CFD, in: AIAA Paper No. 2000-0681*, 38th AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV, 2000.*

[18] D. Poirier, F. Y. Enomoto, The CGNS system, in: AIAA Paper No. 98-3007*, Proceedings of 29th AIAA Fluid Dynamics Conference, Albuquerque, NM, 1998.*

[19] National Laboratory for Scientific Computing (LNCC), Santos Dumont web page, https://sdumont.lncc.br *(Jan. 2019).*

[20] TOP500 Supercomputer List, Laboratório Nacional de Computação Científica, http://www.top500.org/site/50576 *(Dec. 2019).*

[21] H. W. Meuer, E. Strohmaier, J. J. Dongarra, H. D. Simon, The TOP500: History, Trends, and Future Directions in High Performance Computing, 1st Edition, Chapman & Hall/CRC, 2014.

[22] *J. J. Dongarra, Performance of various computers using standard linear equations software, SIGARCH Comput. Archit. News 20 (3) (1992) 22–44.*

[23] *J. J. Dongarra, The LINPACK benchmark: An explanation, in: Proceedings of the 1st International Conference on Supercomputing, Springer-Verlag, London, UK, 1988, pp. 456–474.*

[24] *LNCC, National Laboratory for Scientific Computing (LNCC) web page,* https://www.lncc.br *(Dec. 2019).*

[25] *Lustre, Lustre filesystem page,* https://www.lustre.org/ *(Dec. 2019).*

[26] *RedHat, RedHat web page,* http://www.redhat.com/ *(Dec. 2019).*

[27] *SchedMD, Slurm workload manager page,* https://slurm.schedmd.com/ *(Dec. 2019).*

[28] *A. W. Vreman, Direct and large-eddy simulation of the comperssible turbulent mixing layer, Ph.D. thesis, Universiteit Twente (1995).*

[29] *E. Turkel, V. N. Vatsa, Effect of artificial viscosity on three-dimensional flow solutions, AIAA Journal 32 (1) (1994) 39–45.*

[30] *A. Jameson, D. Mavriplis, Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh, AIAA Journal 24 (4) (1986) 611–618.*

[31] *A. Jameson, W. Schmidt, E. Turkel, Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes, in:* AIAA Paper 81–1259*, Proceedings of the AIAA 14th Fluid and Plasma Dynamic Conference, Palo Alto, Californa, USA, 1981.*

[32] *L. N. Long, M. Khan, H. T. Sharp, A massively parallel three-dimensional Euler/Navier-Stokes method, AIAA Journal 29 (5) (1991) 657–666.*

[33] *J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.*

[34] *J. L. Bentley, Multidimensional binary search trees in database applications, IEEE Transactions on Software Engineering SE-5 (4) (1979) 0–340.*

[35] *P. Balaji, D. Kimpe, On the reproducibility of MPI reduction operations, in: 2013 IEEE 10th International Conference on High Performance Computing and Communications & IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), IEEE, 2013, pp. 407–414.*

[36] *A. Arteaga, O. Fuhrer, T. Hoefler, Desingning a bit-reproducible portable high-performance applications, in: Parallel and Distributed Processing Symposium, 2014 IEEE 28th International, Phoenix, AZ, USA, 2014, pp. 1235–1244.*

[37] *W. Ertel, On the definition of speedup, in: PARLE'94 Parallel Architectures and Languages Europe, Springer, Berlin, 1994, pp. 289–300.*

[38] *X.-H. Sun, Y. Chen, Reevaluating Amdahl's law in multicore era, J. Parallel Distrib. Comput. 70 (2) (2010) 183–188.*

[39] *J. L. Gustafson, Reevaluating Amdahl's law, Communications of the ACM 31 (5) (1988) 532–533.*

[40] *G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: AFIPS Conference Proceedings, Vol. 30, ACM, Atlantic City, N.J., USA, 1967, pp. 483–485.*

[41] *Center for Mathematical Sciences Applied to Industry (CEPID-CeMEAI), Euler computer web page,* https://sites.google.com/site/clustercemeai/ *(Dec. 2019).*

[42] *S. Ristov, R. Prodan, M. Gusev, K. Skala, Superlinear speedup in HPC systems: Why and when?, in: Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, Gdańsk, Poland, 2016, pp. 889–898.*

[43] *M. Gusev, S. Ristov, A superlinear speedup region for matrix multiplication, Concurrency and Computation: Practice and Experience 26 (11) (2014) 1847–1868.*

[44] *X. Jin, T. Yang, X. Tang, A comparison of cache blocking methods for fast execution of ensemble-based score computation, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval,* SIGIR 16*, Pisa, Italy, 2016, pp. 629–638.*

[45] *S. Kamil, P. Husbands, L. Oliker, J. Shalf, K. Yelick, Impact of modern memory subsystems on cache optimizations for stencil computations, in: Proceedings of the 2005 Workshop on Memory System Performance, Chicago, Illinois, 2005, pp. 36–43.*

[46] *D. Gannon, W. Jalby, K. Gallivan, Strategies for cache and local memory management by global program transformation, in: Proceedings of the International Conference on Supercomputing, Athens,Greece, 1987, pp. 229–254.*

[47] *R. Lhner, Cache-efficient renumbering for vectorization, International Journal for Numerical Methods in Biomedical Engineering 26 (5) (2010) 628–636.*

[48] *A. Basermann,* et al.*, HICFD: Highly efficient implementation of cfd codes for hpc many-core architectures, in: Proceedings of an International Conference on Competence in High Performance Computing, Schloss Schwetzingen, Germany, 2010, pp. 1–13.*

[49] *S. Mendez, M. Shoeybi, A. Sharma, F. E. Ham, S. K. L. P. Moin, Large-eddy simulations of perfectly-expanded supersonic jets using an unstructured solver, AIAA Journal 50 (5) (2012) 1103–1118.*

[50] *J. Bridges, M. P. Wernet, Turbulence associated with broadband shock noise in hot jets, in:* AIAA Paper No. 2008-2834*, Proceedings of the 14th AIAA/CEAS Aeroacoustics Conference (29th AIAA Aeroacoustics Conference), Vancouver, Canada, 2008.*

About the Authors

Carlos Junqueira Junior is a Research Engineer at École Nationale Supérieure d'Arts et Métiers (ENSAM), in Paris. He graduated with the Engineering degree at the École Nationale Supérieure de l'Énergie l'Eau et l'Environnement (ENSE3) and also at the Universidade Estadual Paulista (UNESP). He holds the titles of Master of Science and Doctor of Science both achieved at Instituto Instituto Tecnológico de Aeronáutica. His research interests are in the area of computational fluid dynamics, high performance computing, and numerical methods.

João Luiz F. Azevedo is a Senior Research Engineer in the Aerodynamics Division of Instituto de Aeronáutica e Espaço, at São José dos Campos, Brazil. His professional experience includes development and application of CFD codes for applied aerodynamic and aeroelastic analyzes of aerospace vehicles, aeroelastic clearance of launch vehicles, and aerodynamic CFD analyzes of wind tunnel models prior to testing. Areas of research interest include development of high-order, adaptive, unstructured grid CFD codes for realistically complex configurations, implementation of turbulence models, and development of cost effective techniques for coupling CFD solvers with aeroelastic analysis procedures.

Jairo Panetta is a Professor of the Scientific Computing Department at Instituto Tecnológico de Aeronáutica (ITA). He graduated as an engineer and received his Master of Science Degree, both at the Instituto Tecnológico de Aeronáutica. He defended his Ph.D. at Purdue University. His research interests are in the area of computer architecture, analogic processing and high performance computing.

William R. Wolf is a Professor of the Faculdade de Engenharia Mecânica at Universidade Estadual de Campinas (UNICAMP). He holds a Mechanical Engineering degree obtained from the University of São Paulo and the title of Master of Science from Instituto Tecnológico de Aeronáutica. He defended his Ph.D. Thesis at Stanford University. His research interests are in the area of aeroacoustics, turbulent flows, aerodynamics, and computational fluid dynamics.

Sami Yamouni is a Data Scientist at DataLab Serasa Experian. He has an Engineering degree from Institut Supérieur de Mécanique de Paris and a Master of Science degree from the Université de Poitiers. Yamouni defended his Ph.D. Thesis at École Polytechnique. His research interests are in the area of dynamic mode decomposition, proper orthogonal decomposition, artificial intelligence and high performance computing.