Journal of Applied Fluid Mechanics, Vol. 13, No. 4, pp. 1067-1079, 2020.
Available online at www.jafmonline.net, ISSN 1735-3572, EISSN 1735-3645.
DOI: 10.36884/jafm.13.04.30698

PMLES: A Hybrid Open MP CUDA Source Code for LES of Turbulent Flows

J. M. Pinho[1,2†] and A. R. Muniz[2]

[1] *IFSC, Federal Institute of Santa Catarina, Euclides Hack St, 1603, 89820-000, Xanxerê, SC, Brazil*
[2] *UFRGS, Department of Chemical Engineering, Rua Engenheiro Luiz Englert, s/no, 90040-040, Porto Alegre, RS, Brazil*

† *Corresponding Author Email: jean.pinho@ifsc.edu.br*

(Received July 1, 2019; accepted December 28, 2019)

ABSTRACT

The occurrence of turbulent flows is quite common in nature and several industrial applications. The accurate simulation of these complex flows is still a great challenge in science. Large Eddy Simulation (LES) is an efficient technique based on the elimination of all scales of a flow smaller than a characteristic length Δ, considering that the flow pattern in small scales is homogeneous and isotropic. Therefore, modeling of turbulence in such scales is universal and independent of the flow type. This work present PMLES, a new OpenMP CUDA Fortran solver for complex turbulent flows at high Reynolds numbers and large computational domains (about 1×10^8 cells), using a single GPU card. This was possible by using an economical numerical scheme associated with a robust and efficient solution method that requires little variable storage. Theoretical and numerical aspects are firstly discussed, and then details of the computational implementation are given. Finally, the developed code is tested and validated by simulating a turbulent jet, and comparing the results with experimental and computational data from the literature. An analysis of performance gain is also carried out, demonstrating the code's ability to solve this class of problems with a considerable reduction in computational time.

Keywords: CUDA; Open MP; LES; Numerical simulation; Turbulence.

NOMENCLATURE

CFL	Courant-Friedrichs-Lewy condition	$\bar{u}$	mean axial velocity
CFL_{critic}	Critic-Courant Friedrichs-Lewy condition	v_Δ	subgrid characteristic velocity
$C_{i,j}$	cross-stress tensor	x_i	i direction
C_s	Smagorinsky constant	z	instantaneous z-direction velocity
D	nozzle diameter		
Dil	dilatation	$(\sigma_{ij})_{sgs}$	subgrid-scale stress tensor
Dm	domain of filtering operation	$\lvert\bar{S}\rvert$	Frobenius norm
f	generic function	Δ	grid filter
G	filter functions	Δt	time step
H	transverse length of domain	Δt_{conv}	convective time step
L	preferential length of domain	Δt_{dif}	diffusive time step
$L_{i,j}$	Leonard-stress tensor	∇	gradient operator
L_K	Kolmogorov dissipative scale	μ	viscosity
L_λ	length of Taylor micro-scale	μ_e	effective viscosity
N_i	Cells number in i-direction	μ_t	turbulent viscosity
p	pressure	α_i	Runge-Kutta coefficient
r	radial distance of center domain	δ	dimensional filter
Re	Reynolds number	δp	pressure correction
u	instantaneous x-direction velocity	δu_i	i-direction velocity correction
u_i	instantaneous i-direction velocity	ρ	density
U_j	jet velocity	τ_K	characteristic Kolmogorov time
u'	turbulent axial intensity	τ_λ	time of Taylor micro-scale
v	instantaneous y-direction velocity	Φ	time dependent variable
$\bar{u}_{max}$	mean max axial velocity		

1. INTRODUCTION

Turbulence is a phenomenon present in most flows observed in nature and in engineering applications. Turbulent flows are unstable and their properties exhibit fluctuations that are time and space dependent. One of the most striking features of turbulent flows is the multiplicity of scales. They are present from the largest structures (low frequencies), which are controlled by the geometry of the flow, to the smallest structures (high frequencies), which are controlled by molecular viscosity.

Turbulent flows are present in many physical processes and industrial applications, due to the presence of boundary layers and shear flows, such as in turbulent jets, for example. The numerical simulation of turbulent flows remains a challenge in computational fluid dynamics (CFD). In this sense intense research has been made toward understanding and controlling turbulence, due to its importance in a wide variety of engineering applications, such as in aerodynamics, engines, industrial equipment and others.

There are three widely used methods for simulating turbulent flows: Reynolds Averaged Navier-Stokes Simulation (RANS), Large Eddy Simulation (LES) and Direct Navier-Stokes Simulation (DNS). Each one has its own advantages and disadvantages, and the methodology is usually chosen based on the characteristics of the flow and type of the problem to be addressed, as well as of the computational cost and availability of computational resources (Piomelli 1999; Wilcox 2006).

The Large Eddy Simulation (LES) is a efficient technique, based on the elimination of all scales of the flow smaller than a characteristic length scale Δ. The separation of the scales is performed through the proper application of a low-pass filter in the sys-tem of equations, as discussed in the next section. The filtering operations result in the LES equations, able to describe only the flow in the larger scales (Lesieur, Métais and Comte 2005). Conceptually, LES is an intermediate methodology between DNS and RANS, and allow to capture the anisotropic turbulence that occurs in the large scales through the solution of the intermediate scales, while the small scales are described by homogeneous isotropic turbulence models.

The viscous dissipation of the turbulent kinetic energy generated at large scales occurs at smaller scales, called Kolmogorov scales. It is verified that the Kolmogorov scales are smaller than the boundary scales which define the flow (the large scale). It is observed that the behavior of the small scales is little or almost unaffected by the large ones (Tennekes and Lumley 1972). In this sense, it is considered that the flow pattern in small scales is likely homogeneous and isotropic. It is assumed that models for the small scales are more universal, independent of the flow type, when compared to the classical methodology RANS (Piomelli 1999).

The LES technique enables the computational solution of complex transient turbulent flows without a prior knowledge of the behavior of turbulence at the level of resolved scales. However, its computational cost is still far from the acceptable for industrial applications, considering that a LES of a high Reynolds number flow can easily overwhelm the resources of individual workstations.

A possible solution to popularize LES has been consolidating over the last decade, by employing Graphics Processing Units (GPUs) in the calculations. GPUs have been increasingly used to solve scientific problems in many areas. NVIDIA launched its first GP-GPU (General Purpose Computing on Graphics Processing Unit) in 2006, and a parallel programming platform for GPU called CUDA in 2007. Initially CUDA emerged as an expansion of the widely used C language. However, considering that many scientific computational tools had been previously developed in Fortran from a joint work of NVIDIA and PGI (The Port-land Group), a CUDA Fortran Compiler was made available in 2010, which is essentially a conventional FORTRAN compiler with CUDA extensions. After that, many works (Griebel and Zaspel 2010; DeLeon and Senocak 2012; Markesteijn, Semiletov, and Karabasov 2015; Zhu, Phillips, Spandan, Donners, Ruetsch, Romero, Ostilla-M´onico, Yang, Lohse, Verzicco, *et al*. 2018; Kumar, Abdel-Majeed, and Annavaram 2019) have been developed in order to adapt and redesign the existing codes to the architecture of the GPU.

The challenge in implementing codes in CUDA Fortran lies in fundamental differences between CPU and GPU architectures. On-card memory is limited in GPUs, a factor that must be taken into account when designing new GPU algorithms. For example, recomputing certain variables along the simulation can make the code more efficient, compared to computing them once and storing in memory (Markesteijn, Semiletov and Karabasov 2015). According to Markesteijn *et al*, the numerical schemes applied to discretize and solve the equations can make a difference. For optimal computational efficiency, all the computations are desired to be carried out in the GPU using the on-card memory, considering that any copy to conventional (CPU) memory is time consuming. Zhu *et al*. (Zhu, Phillips, Spandan, Donners, Ruetsch, Romero, Ostilla-Mónico, Yang, Lohse, Verzicco, *et al*. 2018) add that GPUs are also characterized by high memory bandwidth, favoring the use of low-order finite differences CFD codes, characterized by minimal data reuse.

One solution to overcome the memory limitation of GPU cards is running the calculations on a cluster of GPUs. However, porting a code originally developed to run in a single GPU to be executed in multiple GPUs may be a complex task. In particular, it requires defining how data is partitioned across multiple GPU cards, and then launch the appropriate thread blocks that can access the local data in each card. Cross-card data transfer is a time-consuming operation and should be avoided when possible (Kumar, Abdel-Majeed and Annavaram 2019). Another alternative to address the memory limitation is to develop a code that is cost-effective in memory usage, able to solve the problem efficiently and with the use of only one GPU. This second approach was used in the present work. The resulting solvers can be run on a single workstation, which represents an interesting advance to-wards bringing LES technique closer to industrial applications.

In this sense the objective of this work is to develop a simple and efficient OpenMP CUDA Fortran solver, named PMLES, to simulate complex turbulent flows capable of simulating high Reynolds numbers and large domains ($\sim 1 \times 10^8$ cells in this work), in relevant geometries of industrial interest using only one GPU card. The use of a single GPU to solve complex problems is achieved by using an economical numerical scheme (low order, but suitable for engineering problems) associated with a robust and relatively cheap numerical method that requires little variable storage. The approach eliminates the overhead of data transfer usually required in MPI-CUDA programming. The code was applied to study the coaxial turbulent jet investigated experimentally by Amielh *et al*. (1996), and the results obtained were satisfactory, considering the quality of the solution and the achieved speedup (55 compared to a serial execution).

This work is organized as follows. Section 2 presents the methodology to obtain the LES equations from the Navier-Stokes equations and modeling their specific terms. In the third section, the numerical methodology to solve the system of the LES equations is discussed. Section 4 is devoted to the proposed implementation methodology, aiming to efficiently utilize the available architecture. In section 5 we describe the test problem and how it was modeled. Section 6 present the results and discussion, and the final section is devoted to the conclusions and future works.

2. LES MODELING

2.1 Filtering Conservation Equations

In the LES methodology, the filtering operation is responsible to separate mathematically the large scales of the flow that will be solved $\bar{f}\,(x,t)$ from the small scales that will be modeled $f'(x,t)$, also called subgrid scales. As a result we have

$$f\,(\mathbf{x},t) = \bar{f}\,(\mathbf{x},t) + f'(\mathbf{x},t) \qquad (1)$$

also known as Leonard decomposition (Pope and Pope 2000). The filtering consists of the convolution of the variable to be filtered with the filter function $\overline{\mathrm{G}}$

$$\bar{f}\,(\mathbf{x},t) \;= \int_0^T \int_{Dm} f\,(\mathbf{x}',t)\ \overline{G}\ \,(\mathbf{x}-\mathbf{x}',: t-t')d\mathbf{x}dt \qquad (2)$$

where *Dm* is the domain on which the operation must be performed.

The filtering process aims to eliminate or smooth out fluctuations that are smaller than the predefined cutoff wave number. However, the filtering reduces the number of degrees of freedom of the problem, which can reduce the precision and the performance of the model. This occurs because there is a decrease in the information contained in the system as the filter size is increased. In contrast, there is a reduction on the computational cost. The challenge is to find a good balance between filter size, accuracy and computational cost.

The LES modeling involves two filtering processes:*i)*a dimensional filter (δ) and *ii)* a grid filter (Δ)(Kuo and Acharya 2012). The phenomena that occurs in a scale smaller than the grid filter cannot be captured by any of the filters, and they are always modeled. The scales smaller than grid filter (Δ) are called sub-grid scales. Figure 1 illustrates the resolved scales and the sub-grid scales.

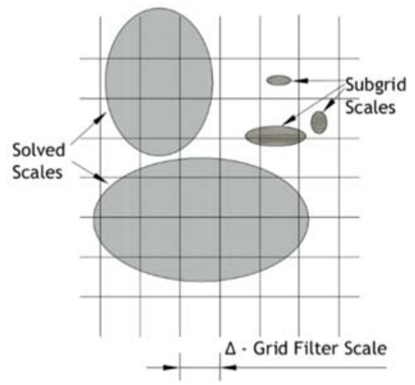

Fig. 1. Resolved and filtered scales in LES methodology.

The Large Eddy Simulation modeling allows using either explicit or implicit filters, provided that they represent the properties of the sub-grid terms. As the mesh is refined, the solution gets closer to the filtered equations when using explicit filters, while for implicit filters the solution approaches the equations obtained by the DNS methodology (Ḧallqvist 2006). Therefore, in LES it does not make sense to analyze mesh convergence. In the limit of grids that allow the smaller scales responsible for viscous dissipation to be captured, more and more scales are solved and fewer scales are modeled. Consequently, the influence of subgrid modeling is diminished and convergence will only occur for results obtained through DNS simulations.

Most applications in LES use the constant volumetric filter, also called top-hat filter (Silva Freire, Menut, and Jian 2002)

$$G(\mathbf{x}) = \begin{cases} 1/\Delta^3\,, \mathrm{se}|x_i| \le \Delta/2, \quad i = 1,2,3; \\ 0, \mathrm{otherwise,} \end{cases} \qquad (3)$$

which is an implicit filter, considering that the characteristic size of the filter is equal to the mesh spacing length. In this case the filtering and differentiation operation commute. This approach is also called Schumann filtering (Huai 2006) and was used in this work.

When the filter function is defined as non-commutative with the differentiation, the filtering is called explicit. The use of explicit filtering has the advantage of clearly separating the size of grid filter (related to the size of the computational cell) from the scale filter (related to the physics of the problem). The other filter functions most commonly used in LES are the Gaussian filter, the cutoff filter (Kuo and Acharya 2012; Piomelli 1999; Veynante and Vervisch 2002). The detailed development of the filtering process is presented in (Kuo and Acharya 2012).

2.2 Favre Average

There are two types of averages that are commonly applied to the conservation equations in the solution of turbulent flows. The first is the Reynolds averaging, that is the conventional temporal average procedure, and the second is the Favre averaging, based on a mass-weighted average. The Reynolds average is widely used for flows with constant density, while the Favre average has been preferred for variable-mass flows, such as in turbulent flames. The Favre averaging is recommended for flows with variable density because the governing equations can be obtained in the same form as those for the incompressible turbulent flow (Kuo and Acharya 2012). (Piomelli 1999) points out that the subgrid terms do not appear in the continuity equation with the use of the Favre averaging. Although the flow under analysis is incompressible and has constant density, the final objective of the code is to study turbulent flows with density variation, and the formulation is desired to be maintained.

A filtered variable *f* with Favre averaging is defined as:

$$\tilde{f} = \frac{\overline{\rho f}}{\bar{\rho}} \tag{4}$$

and the following relations are verified (Kuo and Acharya 2012):

$$\overline{\rho u_i} = \bar{\rho}\tilde{u}_i \tag{5}$$

$$\overline{\rho u_i u_j} = \bar{\rho}\widetilde{u_i u_j} \tag{6}$$

A variable can be then decomposed into its Favre filtered component $\tilde{f}$ and its subgrid component *f*':

$$f\,(\mathbf{x},t) = \bar{f}\,(\mathbf{x},t) + f'(\mathbf{x},t) \tag{7}$$

This procedure can be applied to velocity. Variables whose effects of the density are inherent to the measurement process, such as pressure, stress tensors and the specific mass itself, do not need to be filtered by the Favre average. For these variables, the conventional time averaging can be used (Kuo and Acharya 2012).

2.3 Filtered Conservation Equations

As a result of the filtering process, the momentum equation becomes

$$\frac{\partial(\overline{\rho u_i})}{\partial t} + \frac{\partial(\overline{\rho u_i u_j})}{\partial x_j} = \frac{\partial \bar{p}}{\partial x_i} + \mu\left(\frac{\partial^2 \bar{u}_i}{\partial x_j^2} + \frac{\partial^2 \bar{u}_j}{\partial x_i^2}\right) \tag{8}$$

Details of the application of the filter in the momentum equation can be seen in Moint *et al.* (1991) and Kuo and Acharya (2012).

The nonlinear term of the filtered equation (Eq. 8) resulted in a product of two filtered variables, making their solution unfeasible. This nonlinear term can be treated using the Leonard decomposition in terms of the Favre filter (Sagaut 2006), defined in Eq. 1 and Eq. 4, so that

$$\overline{\rho u_i u_j} = \overline{\bar{\rho}(\tilde{u}_i + u_i')(\tilde{u}_j + u_j')}$$

$$\overline{\rho u_i u_j} = \overline{\bar{\rho}\tilde{u}_i\,\tilde{u}_j} + \overline{\bar{\rho}\tilde{u}_i u_j'} + \overline{\bar{\rho}u_i'\,\tilde{u}_j} + \overline{\bar{\rho}u_i' u_j'} \tag{9}$$

Adding and subtracting the term $\bar{\rho}\tilde{u}_i\tilde{u}_j$ and replacing in the Eq. 8 we have

$$\frac{\partial(\overline{\rho u_i})}{\partial t} + \frac{\partial(\bar{\rho}\tilde{u}_i\,\tilde{u}_j)}{\partial x_j} = \frac{\partial \bar{p}}{\partial x_i} + \mu\left(\frac{\partial^2 \bar{u}_i}{\partial x_j^2} + \frac{\partial^2 \bar{u}_j}{\partial x_i^2}\right)$$
$$-\frac{\partial}{\partial x_j}\left[\overline{\bar{\rho}\tilde{u}_i\,\tilde{u}_j} - \bar{\rho}\tilde{u}_i\tilde{u}_j\right]$$
$$-\frac{\partial}{\partial x_j}\left[+\overline{\bar{\rho}\tilde{u}_i u_j'} + \overline{\bar{\rho}u_i'\,\tilde{u}_j} + \overline{\bar{\rho}u_i' u_j'}\right] \tag{10}$$

The subgrid-scale stress tensor is defined as $(\sigma_{ij})_{sgs}$, as

$$(\sigma_{ij})_{sgs} = \overline{\rho u_i u_j} - \bar{\rho}\tilde{u}_i\,\tilde{u}_j \tag{11}$$

$$(\sigma_{ij})_{sgs} = \bar{\rho}\,(\widetilde{u_i u_j} - \tilde{u}_i\,\tilde{u}_j) \tag{12}$$

$$(\sigma_{ij})_{sgs} = \overline{\bar{\rho}\tilde{u}_i\,\tilde{u}_j} - \bar{\rho}\tilde{u}_i\,\tilde{u}_j + \bar{\rho}(\overline{\tilde{u}_i u_j'} + \overline{u_i'\tilde{u}_j})$$
$$+\overline{\bar{\rho}u_i' u_j'} \tag{13}$$

$$(\sigma_{ij})_{sgs} = L_{i,j} + C_{i,j} + R_{i,j} \tag{14}$$

where $L_{i,j}$ is the Leonard-stress tensor that represents the interaction between the resolved scales, that result in the subgrid contributions, $C_{i,j}$ is the Cross-stress tensor that represents the interaction between the resolved scales and the unresolved scales, and $R_{i,j}$ is the Reynolds-stress tensor that represents the interaction between the unresolved small scales.

$$\frac{\partial\bar{\rho}}{\partial t} + \frac{\partial\bar{\rho}\tilde{u}_i}{\partial x_i} = 0 \tag{15}$$

$$\frac{\partial(\bar{\rho}\tilde{u}_i)}{\partial t} + \frac{\partial(\bar{\rho}\tilde{u}_i\,\tilde{u}_j)}{\partial x_j} = -\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial(\sigma_{ij})_{sgs}}{\partial x_i} +$$
$$\frac{1}{Re}\left(\frac{\partial^2\tilde{u}_i}{\partial x_j^2} + \frac{\partial^2\tilde{u}_j}{\partial x_i^2}\right) \tag{16}$$

2.4 Subgrid Stress Tensor Modelling

In the framework of LES, there are many submodels for describe the subgrid stress tensor. The studies of Piomelli (1999), Lesieur *et al.* (2005) and Sagaut (2006) discuss these models in detail. The ideal model for the subgrid stress tensor should provide a correct description of the interaction between the resolved scales and unresolved scales, describing the flow of kinetic energy between the scales in both directions (forward and reverse energy cascades). However, most of the models currently used consider only the energy flow of the large scales for the small scales and do not consider the reverse energy transfer, which has a considerably lower intensity (Sagaut 2006). The most commonly used is the Smagorinsky Model (Smagorinsky 1963), based on the concept of turbulent viscosity proposed by Boussinesq.

The concept of eddy viscosity introduces the following hypothesis: “the energy transfer mechanism of the scales solved to the subgrid scales is analogous to the molecular momentum transfer mechanism, represented by the diffusive term with the viscosity μ". According to this principle, the subgrid stress tensor can be described as:

$$\sigma_{ij}^{sgs} = -\nu_t\left(\frac{\partial\tilde{u}_i}{\partial x_j}+\frac{\partial\tilde{u}_j}{\partial x_i}\right)+\frac{2}{3}\nu_t\frac{\partial\tilde{u}_k}{\partial x_k}\delta_{ij} \tag{17}$$

This definition (Eq. 17) does not include the contribution of the isotropic portion, which can be done by using a modified static pressure. However, as the study of acoustic interactions and compressibility effects are out of the scope of this work, this isotropic contribution will be neglected, and as a consequence, there is an increase on the computational efficiency (Pierce and Moin 2004; Pierce 2001).

The use of the subgrid stress tensor as defined in Eq.17 is practical because it allows combining the eddy viscosity μ_t with the molecular viscosity, resulting in a dimensionless effective viscosity μ_e in the numerator of the diffusive term of the momentum conservation equation (Huai 2006):

$$\mu_e = \frac{\mu+\mu_t}{\mu}, \tag{18}$$

Then, the Eq. 16 can be rewritten as

$$\frac{\partial(\bar{\rho}\tilde{u}_i)}{\partial t}+\frac{\partial(\bar{\rho}\tilde{u}_i\tilde{u}_j)}{\partial x_j} = -\frac{\partial\bar{p}}{\partial x_i}+$$

$$+\frac{\mu_e}{Re}\left(\frac{\partial^2\tilde{u}_i}{\partial x_j^2}+\frac{\partial^2\tilde{u}_j}{\partial x_i^2}\right) \tag{19}$$

Although it is known that $\mu_t \gg \mu$, keeping both terms enhance numerical stability, avoiding null values for these properties. Besides, Kuo and Acharya (2012) claims that molecular transport effects may be important near the walls and close to turbulent/non-turbulent interfaces (TNTI).

2.5 Smagorinsky Model

The turbulent viscosity which arises in the model described in the previous section still needs to modeled. In this work, the turbulent viscosity will be described by the model of Smagorinsky (1963). Smagorinsky (1963) assumed that the eddy viscosity (μ_t) is proportional to the characteristic length of the filter Δ, and to the characteristic subgrid velocity v_Δ, that are defined as

$$v_\Delta = \Delta|\bar{S}| \tag{20}$$

$$\Delta = \sqrt[3]{\Delta x\Delta y\Delta z} \tag{21}$$

where the norm $|\bar{S}|$ is calculated from the second invariant of strain rate tensor.

$$|\bar{S}| = \sqrt{2\bar{S}_{ij}\bar{S}_{ij}} \tag{22}$$

Then, the turbulent viscosity is evaluated as

$$\mu_t = \rho(C_s\Delta)^2|\bar{S}|, \tag{23}$$

where C_s is called the Smagorinsky constant. We can see that the turbulent viscosity has a quadratic dependence on C_s; large values of C_s may then introduce significant dissipation on the model, being able to kill the turbulence due to an excess of dissipation of the turbulent kinetic energy in the small scales modeled. On the other hand, a small value for C_s makes the solution procedure unstable, considering that the turbulent kinetic energy produced at large scales and transported to the small scales is dissipated at a lower rate than it is being generated. Consequently, the hypothesis of local equilibrium of turbulent kinetic energy, for larger scales than the dissipative scale of Kolmogorov, is not respected (Hällqvist 2006).

The positive characteristics of the Smagorinsky model for subgrid stress tensor are the easy implementation, low computational cost and satisfactory results for a large number of engineering applications. On the other hand, their main weaknesses include the excessive dissipation near surfaces, not reproducing the reverse energy cascade, and the necessity of defining/choosing the ad hoc (C_s) constant, according to the flow characteristics.

For turbulent jets, there is no consensus in the literature about the value of C_s than should be used, and a wide range has been used. The theoretical value is for Smagorinsky constant is 0.18 (Pope and Pope 2000; Sagaut 2006). Ylyushin and Krasinsky (2006) used C_s = 0.17, as suggested by Pope and Pope (2000); Wilson and Demuren (1997), and Jones *et al*. (2002) used 0.1-0.12 as suggested by Lesieur (2005) while Deardorff (1970), McMillan (1980) and Ferziger and Peric (2012) use C_s between 0.065-.1. A study for the optimal value of C_s in turbulent jets using the PMLES code will be carried out in a future work.

3. Numerical Methodology

The filtered equations presented above (Eq. 19) are discretized by the finite difference method, using second-order centered schemes for inner mesh points and backward or forward second-order schemes for the boundary points. This numerical scheme is simple, inexpensive, non dissipative, and have reasonable accuracy for engineering problems.

The present version of the code is able to work only for regular three-dimensional structured cartesian grids, with a uniform spacing between points, $\Delta x = \Delta y = \Delta z$. Despite of the apparent simplicity of this type of mesh, it has some interesting features. Implicit filtering is used as described in Sec. 2.1, and the use of a regularly spaced mesh avoids the propagation of errors due to filter size variations, as described by Piomelli (1999) and shown by Ilyshin and Kransinky (2006). Variable integration timesteps are used, applying the stability condition of Courant Friedrichs-Lewy (CFL), evaluated as described in Ferziger and Peric (2012) and performed in Damasceno *et al.* (2015) by

$$\Delta t = CFL_{critic}\left(\frac{1}{\Delta t_{conv}}+\frac{1}{\Delta t_{dif}}\right)^{-1} \tag{24}$$

Where

$$\Delta t_{conv} = \left(\sum_{i=1}^{3}\frac{\Delta x_i}{|u_i|\max}\right) \tag{25}$$

$$\Delta t_{dif} = \left(\sum_{i=1}^{3} \frac{\Delta x_i^2}{\mu_e}\right) \tag{26}$$

Equation (19) was discretized in time using the three-stage second-order Runge-Kutta scheme presented by Blazek (2015):

$$\begin{aligned}
\phi^0 &= \phi^n \\
\phi^1 &= \phi^0 + \alpha_1 \Delta t * R^0(\phi) \\
\phi^2 &= \phi^0 + \alpha_2 \Delta t * R^1(\phi) \\
\phi^3 &= \phi^0 + \alpha_3 \Delta t * R^2(\phi)
\end{aligned} \tag{27}$$

where ϕ is the time dependent variable and $R(\phi^n)$ correspond to the terms that not include the time derivative, such as source terms and discretized spatial derivatives. α_m are the coefficients of each stage, given by $\alpha_1 = 0.1918$, $\alpha_2 = 0.4929$, $\alpha_3 = 1$, where for these coefficients $CFL_{critic} = 0.65$. This group of explicit schemes for time integration is computationally cheap, consumes little memory and can be employed with any spatial discretization scheme. The use of an explicit scheme, when working on a SIMT (Single Instruction Multiple Thread) architecture (of GPU cards) (Quadros 2016), makes the method quite interesting, because the architecture of the GPU enables the massively parallel execution of thousands of threads independently and simultaneously (Ruetsch and Fatica 2011).

Although the equations of flow are presented herein in compressible form (aiming to facilitate future developments), the present version of the PMLES code is only able to simulate flow at low Mach numbers. In this case, the continuity equation does not have a dominant variable, and it configures itself as a kinematic constraint that the velocity field must respect (Ferziger and Peric 2012).

Here the calculation of the pressure field for incompressible flows is performed using the SOLA (SOLution Algorithm) method (Hirt, Nichols, and Romero 1975; Wilson, Nichols, Hirt, and Stein 1988; Fortuna 2000) which consists in an iterative procedure to correct the pressure on a given mesh point at an timestep n + 1

$$p_{(i,j,k)}^{n+1,k+1} = p_{(i,j,k)}^{n+1,k} + \delta p_{(i,j,k)}^{n+1,k+1}, \tag{28}$$

where

$$\delta p_{(i,j,k)}^{n+1,k+1} = \frac{-Dil^{n+1,k+1}}{2\Delta t\left[\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\right]} \tag{29}$$

and

$$Dil = \nabla.(\rho \tilde{u}_i). \tag{30}$$

In this method, the pressure of a cell is increased/decreased if there is a net mass flow in-ward/outward the cell. The flow direction analysis and the computation of the pressure correction δp are done using the Dilatation *Dil,* defined in Eq. 30. The velocity field is then corrected using the pressure correction according to the following equations:

$$\delta u_{(i+\frac{1}{2},j,k)}^{n+1,k} = \frac{\Delta t}{\rho_{(i+\frac{1}{2},j,k)}^{n}} \frac{\delta p_{(i,j,k)}^{n+1,k}}{\Delta x} \tag{31}$$

$$\delta u_{(i-\frac{1}{2},j,k)}^{n+1,k} = -\frac{\Delta t}{\rho_{(i-\frac{1}{2},j,k)}^{n}} \frac{\delta p_{(i,j,k)}^{n+1,k}}{\Delta x} \tag{32}$$

$$\delta v_{(i,j+\frac{1}{2},k)}^{n+1,k} = \frac{\Delta t}{\rho_{(i,j+\frac{1}{2},k)}^{n}} \frac{\delta p_{(i,j,k)}^{n+1,k}}{\Delta y} \tag{33}$$

$$\delta v_{(i,j-\frac{1}{2},k)}^{n+1,k} = -\frac{\Delta t}{\rho_{(i,j-\frac{1}{2},k)}^{n}} \frac{\delta p_{(i,j,k)}^{n+1,k}}{\Delta y} \tag{34}$$

$$\delta w_{(i,j,k+\frac{1}{2})}^{n+1,k} = \frac{\Delta t}{\rho_{(i,j,k+\frac{1}{2})}^{n}} \frac{\delta p_{(i,j,k)}^{n+1,k}}{\Delta z} \tag{35}$$

$$\delta w_{(i,j,k-\frac{1}{2})}^{n+1,k} = \frac{\Delta t}{\rho_{(i,j,k-\frac{1}{2})}^{n}} \frac{\delta p_{(i,j,k)}^{n+1,k}}{\Delta z} \tag{36}$$

The pressure and velocity fields are recursively corrected by the above equations until they reach convergence. Hirt *et al.* (1975) and Fortuna (2000) give a detailed explanation about the method and its implementation.

Analyzes of LES performed with different subgrid models have shown that in many cases they do not have a significant influence on the accuracy of the solution, but the proper description of the boundary conditions do (Ilyushin and Krasinsky 2006). The works of Tabor and Baba-Ahmadi (2010), Montorfano *et al.* (2013) and Damasceno *et al.* (2015) present a good discussion about the LES inlet boundary condition modeling. In the present version of the code, boundary conditions similar as those used in simulations of laminar flows were employed, and the computational domain at the inlet is then extended to allow the development of the flow instabilities, reducing the influence of the inlet boundary on results. More specifically, the inlet boundary and the rigid boundary walls are modeled by Dirichlet conditions, and a Neumann boundary condition for fully developed flows was used at the outlet, assuming that the gradient of normal momentum flux is null. Future versions of PMLES will include other possibilities of defining the inlet boundary conditions.

4. IMPLEMENTATION METHODOLOGY

CUDA Fortran programming is a hybrid programming model; the control of the execution flow is done by the host, which can also execute subprograms and functions, while parts of code are executed by the GPU (device). The goal of the programmer is to partition the program into blocks of great granularity, which can be executed in parallel. Each block will then be partitioned into others of smaller granularity, to be executed in parallel by CUDA cores.

The developed code - PMLES - is a OpenMP CUDA Fortran solver. Figure 2 depicts a flowchart for the algorithm implemented in PMLES. Blocks colored in blue and green correspond to identified tasks performed by the host and the device (GPU), respectively. It should be noted that all the arrows that connect the green boxes (GPU kernels) are blue. It means that all flux control and some necessary

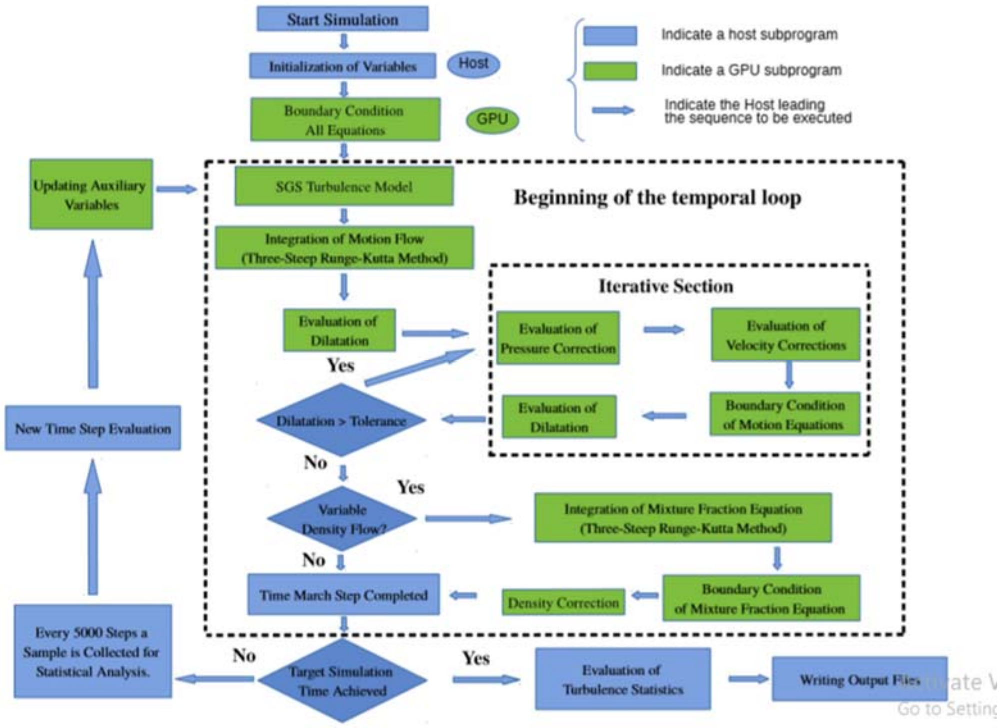
Start Simulation
Initialization of Variables
Host
Boundary Condition All Equations
GPU
Indicate a host subprogram
Indicate a GPU subprogram
Indicate the Host leading the sequence to be executed
Updating Auxiliary Variables
SGS Turbulence Model
Beginning of the temporal loop
Integration of Motion Flow (Three-Steep Runge-Kutta Method)
Iterative Section
Evaluation of Dilatation
Evaluation of Pressure Correction
Evaluation of Velocity Corrections
Yes
New Time Step Evaluation
Dilatation > Tolerance
Evaluation of Dilatation
Boundary Condition of Motion Equations
No
Yes
Variable Density Flow?
Integration of Mixture Fraction Equation (Three-Steep Runge-Kutta Method)
No
Time March Step Completed
Density Correction
Boundary Condition of Mixture Fraction Equation
Every 5000 Steps a Sample is Collected for Statistical Analysis.
No
Target Simulation Time Achieved
Yes
Evaluation of Turbulence Statistics
Writing Output Files

Fig. 2. PMLES Solver Flowchart.

synchronizations are performed by the host, thus characterizing it by a hybrid execution, on which the GPU cards are commanded by the CPU.

The initial tasks of declaration, allocation (both on the host and device), and initialization of variables are performed on the host. After the variables initialization, the host invokes the boundary conditions kernels and starts the temporal loop. In the temporal loop, the host calls the sequence of GPU kernels that are executed on the CUDA cores, i.e., the kernel of the Sub-Grid-Scale (SGS) Turbulence Model, followed by the kernels of the integration of momentum equations on each direction (Eq. 19) and of the Dilatation equation (Eq. 30), finally entering the iterative section described from Eq. 28 to Eq. 36, which runs until convergence is reached. If the flow has variable density (from the mixture of different fluids, for example), the mixture fraction equation is integrated and density is updated. The analysis of non-reactive jet mixture flows and reactive jet flows (both with varying density) will be presented in a forthcoming publication.

The flowchart in Fig. 2 shows that the most intensive tasks of the solver are implemented in CUDA Fortran, namely the subprograms which perform loops over the whole domain. We may also note that some subprograms (setting boundary conditions, for example), are executed on the device, even when it would be cheaper to run them on the CPU. This approach follows an important good practice suggested by (Ruetsch and Fatica 2011) - perform as many GPU operations as possible in order to minimize data transfer between GPU and CPU, since such transfers are usually expensive and greatly penalizes the execution time.

However some operations are not simple to perform in GPU. In the code presented in the article, OpenMP directives are used to evaluate the control variables of pressure correction loop. The maximum values of variables such as dilatation and effective viscosity are determined using OpenMP (reduction operations) directives, resulting in a considerable gain, since the computational domain is large. There are also subprograms that run occasionally on the host, and are programmed in OpenMP, such as writing output files and collecting samples for statistical analysis of the solution. These subprograms run on the host using OpenMP directives characterize PMLES as an “OpenMP CUDA Fortran code”.

Since an important limitation of the GPU card architecture is the available memory, the implementation was focused on memory saving. Due to the features of the present numerical method, some care must be taken with respect to the synchronism and execution sequence of kernels on the device.

On CUDA Fortran programming we can create threads called CUDA streams. The use of CUDA streams allows kernels to be launched for simultaneous execution on the device. This possibility, combined with an adequate definition of the size of the thread blocks, results in a high GPU occupancy rate, reducing code execution time. The optimal thread block size should be set according to the characteristics of the GPU card and the loop boundaries, as described in (Ruetsch and Fatica 2011). Another benefit of using CUDA streams is the ability to perform data transfer (device to host when needed) or update variables concurrently with kernels executions.

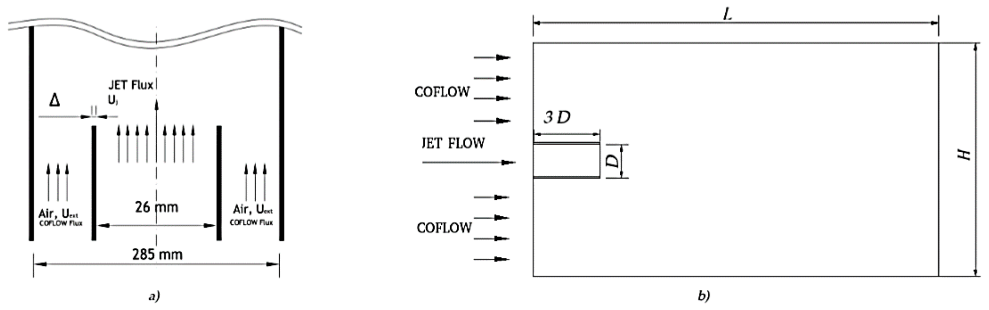

Fig. 3. a) Characteristic dimensions of the nozzle region. b) Details of computational domain geometry.

The kernels responsible for integrating the momentum equations and for evaluating boundary conditions (except for inlet and outlet boundary) can be executed concurrently. Other kernels or sub-programs executed on the GPU cannot be executed concurrently due to the required synchronism, which is responsible for the separation of subprograms out of the iterative part of the code shown in Fig. 2.

5. Description of the Test Problem and Computational Domain

The implemented code must be tested and validated against reliable results previously published. The choice of the test problem followed the guide-lines of Morgans *et al.* (1999): *a)* availability of appropriate experimental data (e.g. boundary distributions of velocity and turbulence quantities);*b)* observed or deduced sensitivity of the flow to changes in boundary conditions; *c)* feasibility of obtaining the numerical solution. Based on this strategy, the experimental turbulent jets studies of of Amielh *et al.* (1996) and Djeridane *et al.* (1996) were chosen. They provide detailed information on velocity and scalar profiles for turbulent jets. The experimental scheme is depicted on Fig. 3a. The authors classify this configuration as a slightly confined jet, and the $(Air - Air)$ jet was used as their reference case and will be the object of our study. This experiment set have been also realized for jets with variable density ratios,($s = \rho_j/\rho_{air}$) 0.14 ($He-Air$) and 1.52 ($CO_2 - Air$) using the same configuration. A comparison with these results will be done in a future work.

The knowledge of the turbulent scales is of great importance in the analysis of turbulent flows. As briefly discussed in previous sections, all the conceptual and mathematical development of LES methodology is derived from the analysis and study of turbulent scales. The characteristic numbers of the flow were determined from the Reynolds number, $Re = \frac{\rho U_j D}{\mu} = 20650$, where $U_j = 12$ m/s is the jet flow velocity, and the length scale is the nozzle diameter D, illustrated in Fig. 3a. The air properties have been taken to standard condition. The characteristic length and time of the dissipative scale of Kolmogorov are given by

$$L_K = \frac{L}{Re^{\frac{3}{4}}} = 5.80 \times 10^{-4}, \tag{37}$$

$$\tau_K = \frac{D/U_j}{Re^{\frac{1}{2}}} = 6.98 \times 10^{-3}, \tag{38}$$

and the characteristic length and time of the Taylor's micro scale by

$$L_\lambda = \frac{L\sqrt{10}}{Re^{\frac{1}{2}}} = 2.2 \times 10^{-2}, \tag{39}$$

$$\tau_\lambda = \frac{\sqrt{15}(D/U_j)}{Re^{\frac{1}{2}}} = 2.7 \times 10^{-2}, \tag{40}$$

The dissipative Kolmogorov scale indicate the mesh resolution for a DNS, while Taylor's micro scale provides an estimate of the mesh resolution suitable for LES (Sagaut 2013).

The domain used to define the problem described above (Amielh *et al.* 1996; Djeridane *et al.* 1996) is a rectangular duct section with a circular coaxial duct for the high velocity fluid injection. A plane view is shown in Fig. 3b. The original setup has dimensions $L \simeq 50D$ and $H \simeq 11D$ (Amielh *et al.* 1996; Djeridane *et al.* 1996). Due to the high computational cost of the full problem, the analysis is focused on regions near the injector, called near field. The near field comprises the zone of pure jet, dominated by inertial effects and by the transition zone, where the inertial effects and gravitational coexist (the latter in the presence of fluids with different densities). The analysis of the developed zone is outside the scope of this work. The three jets zones, as classified for Lipary and Stansby (Lipari and Stansby 2011) - pure jet, transitional and developed zones - can be pictorially visualized in Fig. 4.

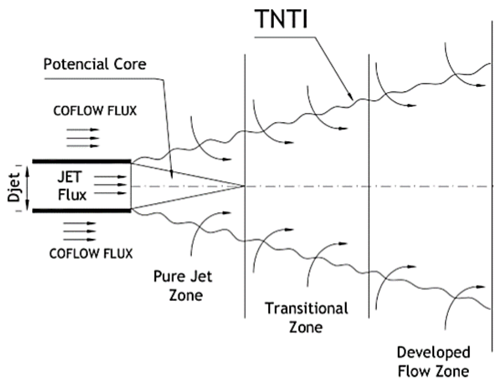

Fig. 4. Depiction of typical jet zones.

Therefore here the domain was limited to $L \simeq 35D$ and $H \simeq 11D$, based on the analysis of experimental data of Amielh *et al*. (1996). The computational domain was discretized using N_x = 1004, N_y = 317 and N_z = 317, resulting on a mesh spacing of $\Delta x = 3.49 \times 10^{-2}$ and a domain with 100890956 ~ 1×10^8 cells. Despite using a Cartesian mesh, the circular injection duct of the jet was relatively well modeled, as shown in Fig. 5. The momentum flux deviation by the cubic cells was evaluated and remained below 1 *%*.

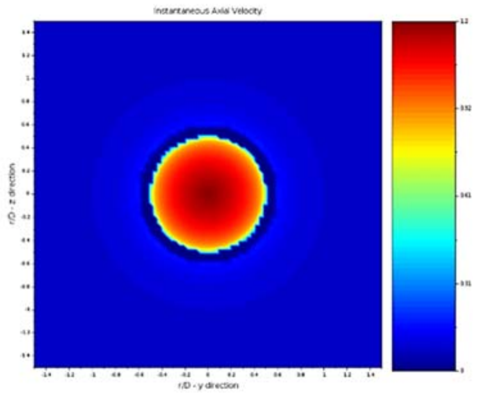

Fig. 5. Circular nozzle as modeled by cubic cells.

As discussed in Sec. 3, Dirichlet boundary conditions are used for the inlet boundary and duct walls in the momentum equations. Considering that the walls are solid and impermeable, non-slip conditions (null velocities) are applied in all walls. For both the COFLOW and JETFLOW boundaries, average speed profiles were used without fluctuations. A flat velocity profile was applied for the COFLOW region, while a fully developed turbulent profile was used for the JETFLOW boundary, which can be evaluated for a circular duct by the following power-law expression (Abramovich 1963).

$$\frac{\bar{u}(r)}{\bar{u}_{max}} = \left(1 - \frac{r}{R}\right)^{1/7}, \tag{41}$$

where $\bar{u}$ *(r)* is the is the velocity of the flow at a distance r from the center of the injector and $\bar{u}_{max}$ is the maximum jet velocity. For the outlet boundary, a Neumann boundary condition was used for fully developed flows, assuming that the gradient of normal momentum flux is null.

The subgrid stress tensor is modelled by subgrid Smagorinsk Model. For the ad hoc constant the Smagorinsk Model, C_s, we set the value of 0.65, as suggested by Ferziger and Peric (2012).

6. RESULTS

The main objective of the work is to present the mathematical basis and computational aspects used in the development of the PMLES source code. As discussed before, the code is tested and validated using a classical problem from the literature. Results obtained for a typical simulation of a air jet flow, as described in details in the previous sections, is presented in this section.

Figure 6 shows a view of the instantaneous velocity field of a turbulent flow generated by the code, illustrating the capture of the transition to turbulence. The arising of the Kelvin-Helmholtz instabilities and their evolution toward turbulence can be clearly seen in the image. Other important result is presented in Fig. 7, which shows the mean velocity field. In this figure, the occurrence of the three characteristics zones of round jets (pure jet, transitional flow and developed flow) is evident as also depicted in Fig. 4 and presented by Lipary and Stansby (Lipari and Stansby 2011). The results depicted in Figs. 6 and 7 show qualitatively the ability of the code to simulate turbulent flows.

A quantitative evaluation of results can be done by analyzing velocity and turbulent intensity profiles at important sections of the flow. Figure 8a shows a comparison of the average axial velocity profile with the experimental data of Amielh *et al*. (1996), LES results of Wang *et al*. (2008) and the similarity law proposed by Chen and Rodi (1980). This profile corresponds to an average on the statis-tically steady-state regime, assumed when the mean velocity component in the preferred flow direction stopped varying significantly for three consecutive samples (taken every 2×10^6 timesteps).

The profiles presented on Fig. 8a have a very good agreement with the other data, exhibiting some localized discrepancies. In this figure, we can observe a small deviation near the nozzle, within the pure jet and transitional zones, as defined in Fig.4. The deviation is in the same order of magnitude of those obtained by Wang *et al*. (2008). Our result also presented good agreement with the law similarity proposed by Chen and Rodi (1980). The comparison with the law similarity is important, considering that it is derived from the study of many turbulent jets, representing an overall behavior of such flows.

The delay in the transition of jet flow regime verified in the Fig. 8a was expected, and it is due to the application of the simplified inlet boundary condition for the jet described in Sec. 6 (in terms of the average velocity, without fluctuations). The implementation of a more consistent inlet boundary condition (as discussed in Sec. 6) will be done and shown in a future work, enabling the simulation of more realistic flows.

Figure 9 show radial profiles for the preferential velocity component at different axial distances *x/D*. In these figures we can observe that the code captures qualitatively well the behavior of the radial distribution of axial velocity, and some deviations between the simulated and experimental values are evident from these figures. These differences are consistent with the results observed in Fig. 8a (dis-cussed in the previous paragraph). The profile in *x/D* = 5 of Fig. 9a shows clearly the delay in the decay of the axial velocity for the zone between the pure jet and transitional flow regimes. For *x/D* = 5, the velocity distribution along the radial direction is still similar to the profile defined in the boundary condition at the nozzle. It can also be observed that, as the profile is measured further away the nozzle, the deviations with the experimental data decrease

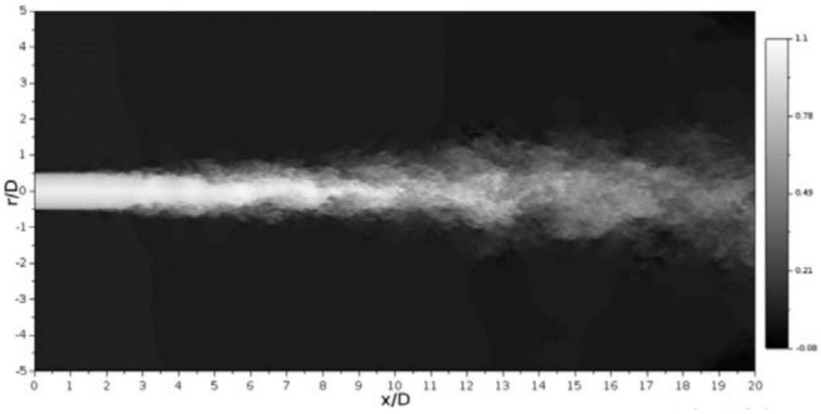

Fig. 6. Visualization of instantaneous field of axial velocity.

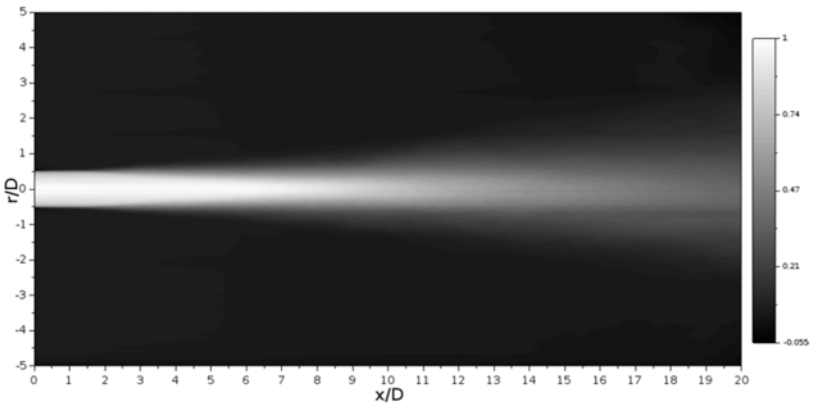

Fig. 7. Visualization of mean field of axial velocity.

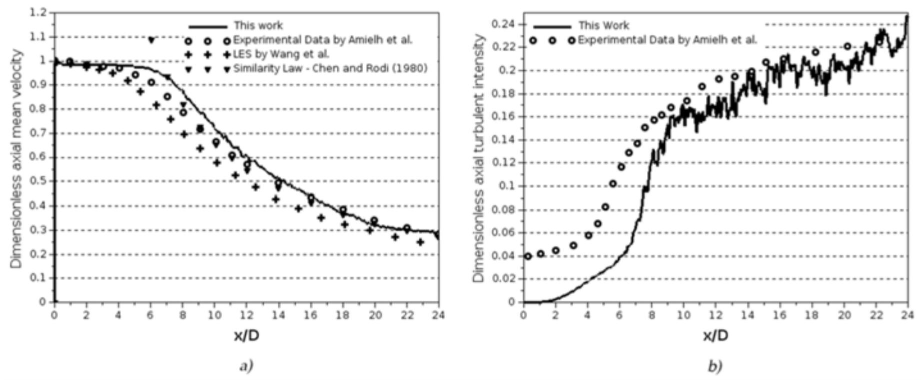

Fig. 8. **a) Dimensionless mean axial velocity in the center line of domain, defined as $(U_{center} - U_{coflow})/(U_{jet} - U_{coflow})$. b) Axial profile of dimensionless turbulent intensity, u' , defined by u' /$(U_{center} - U_{coflow})$.**

Figure 8b shows an important result that LES is able to provide, which is the intensity of the velocity fluctuations, here only analyzed for the preferential direction of the flow. A excellent qualitative agreement is verified, with a systematic deviation in the pure jet zone, due to the simplified boundary condition as discussed before. As turbulence develops, the deviation decreases and computed values get closer to the experimental result. This result clearly shows the PMLES ability of solving the turbulence transition on such flows.

It is naive to attribute any and all deviation in the results simply to the inlet boundary condition used for the jet, given the known limitations of the Smagorinsky subgrid model. However, considering that the largest deviations observed are in regions close to the nozzle, and that they decrease upon development of the turbulence, it is clear that the adequate modeling of this boundary condition deserves attention before carrying out a more comprehensive quantitative comparison of results.

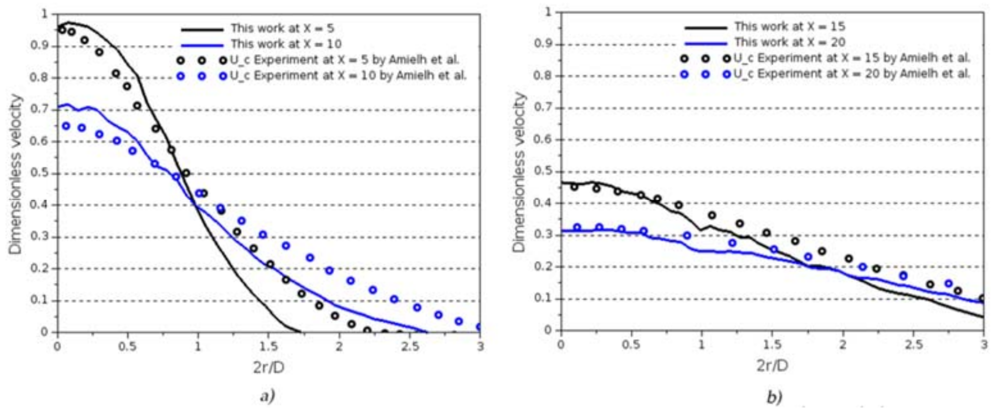

Fig. 9. Dimensionless radial profile of mean axial velocity defined by $(U_{center} - U_{coflow})/(U_{jet} - U_{coflow})$. a) For x/D = 5 and x/D = 10; b) for x/D = 15 and x/D = 20.

The simulations using the hybrid OpenMP CUDA Fortran code were performed on the SDumont supercomputer, available at the LNCC - Laboratorio Nacional de Computacão Científica, Brazil. Although the GPUs used have support for OpenACC directives, specific kernels were developed for the solution of the flow equations in order to maximize the performance of the available computational resource. The solver was not designed with portability in mind, but rather to simulate large domains using few computational resources. Therefore, the simulations were performed using only one computational node, which has two 12-processor IN-TEL XEON E-52695V2 CPUs and two NVIDIA TESLA K40 GPUs, where each one has 2280 CUDA cores with base clock of 745 MHz and 12 GB of DDR5 memory.

Table 1 shows the computational performance gain per timestep (in the problem analyzed in this work) for the parallelization techniques used. The execution in CUDA utilized only a single GPU card as discussed before. The performance enhancement obtained with OpenMP CUDA is interesting; a speedup of 55.1 has been achieved for this ma-chine configuration and the execution time is reduced to about 1/3 compared to a pure OpenMP implementation.

Table 1 Speedup obtained for each parallelization technique

Code Version	time(s)	speedup
Serial	8.90	1
OpenMP with 12 threads	1.04	8.5
OpenMP with 24 threads	0.53	16.6
OpenMP CUDA	0.16	55.1

7. CONCLUSIONS AND FUTURE WORKS

This work presented PMLES, a new computational tool based on a hybrid implementation OpenMP CUDA Fortran, able to simulate complex turbulent flows at high Reynolds numbers with reasonable time costs, using a workstation with a single GPU card. The code was tested in the simulation of an air-air coaxial turbulent jet previously studied in the literature (Amielh *et al.* 1996; Wang *et al.* 2008). The results obtained for the axial distributions of the axial velocity component, in terms of both the mean and the turbulence intensity, are of high quality considering the complexity of the problem solved.

Some of the future improvements in the capabilities of the code are the implementation of a turbulent boundary condition and the implementation of other SGS models, such as the Germano dynamic model (Germano *et al.* 1991) and the structure-function (SF) model (Métais and Lesieur 1992), to eliminate the ad hoc (C_S) constant of the Smagorinsky sub grid model, making the solver more robust and generic. Also, we are working to adapt the code to simulate turbulent jets of variable density, a problem that requires not only the modeling of the tensor stresses, but also the modeling of the scalar flux. Also, this current version of PMLES still lacks the ability to properly solve turbulent flows on surfaces, such as in aero-dynamics. This restriction is due to the occurrence of boundary layers, which require special treatment due to the failure of the hypothesis of homogeneous and isotropic flow, which is the basis of the LES technique. An adequate treatment for flows near the walls will soon be implemented.

The computational performance obtained with the use of the GPU card was very good, since it increased the speedup by a factor of 55.1, compared to 16 obtained by Griebel and Zaspel (2010) and 33 obtained by Thibault and Senocak (2009), decreasing the time by 3.3× with respect to the solution obtained using full OpenMP.

A limitation of the present implementation, employing only one GPU card, is the limited memory of the GPU device, which is small compared to that typical of CPUs. This memory limitation imposes a constraint on the size of the grid, i.e., in the degree of mesh refinement and domain size. However, with the growing development of new technologies for scientific computing, this limitation is being reduced. In 2019 there are GPU cards in the market capable of simulating domains of the order of 500 million cells using PMLES. To overcome the memory limitation, multiple GPUs could be used, keeping in mind that there are extra costs involved in the data transfer between GPUs, especially if it is necessary to perform communication within the iterative block of code.

The speedup analyzes shown in Table 1 are all performed for calculations with double precision, which is the least favorable condition to gain speed with the use of GPU. There is now in the literature mixed precision implementations for computational fluid dynamics that present considerable reduction of computational cost delivering a reasonable acceptable accuracy for engineering applications. A mixed precision implementation will be analyzed as well in future works.

ACKNOWLEDGMENTS

The authors acknowledge CAPES for financial support through PROEX/CAPES program, and the National Laboratory for Scientific Computing (SDumont supercomputer, LNCC/MCTI, Brazil) for providing computational resources for the calculations reported in this paper.

REFERENCES

Abramovich, G. (1963). *The theory of turbulent jets*. Ph. D. thesis, MIT Press, Massachusetts Institute of technology.

Amielh, M., T. Djeridane, F. Anselmet and L. Fulachier (1996). Velocity near-field of variable density turbulent jets. *International Journal of Heat and Mass Transfer* 39(10), 2149–2164.

Blazek, J. (2015). *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann.

Chen, C. J. and W. Rodi (1980). Vertical turbulent buoyant jets: a review of experimental data. NASA Sti/Recon Technical Report A 80.

Damasceno, M., J. Vedovoto and A. da Silveira-Neto (2015). Turbulent inlet conditions modeling using large-eddy simulations. *Computer Modeling in Engineering & Sciences* 104(2), 105–132.

Deardorff, J. W. (1970). A numerical study of three-dimensional turbulent channel flow at large reynolds numbers. *Journal of Fluid Mechanics* 41(2), 453–480.

DeLeon, R. and I. Senocak (2012). Gpu-accelerated large-eddy simulation of turbulent channel flows. *In 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pp.722.

Djeridane, T., M. Amielh, F. Anselmet and L. Fulachier (1996). Velocity turbulence properties in the near-field region of axisymmetric variable density jets. *Physics of Fluids* 8(6), 1614–1630.

Ferziger, J. H. and M. Peric (2012). Computational methods for fluid dynamics. *Springer Science & Business Media.*

Fortuna, A. D. O. (2000). *Técnicas computacionais para dinâminca dos fluidos: conceitos básicos e aplicacões. Edusp.*

Germano, M., U. Piomelli, P. Moin and W. H. Cabot (1991). A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics* 3(7), 1760–1765.

Gharbi, A., E. Ruffin, F. Anselmet and R. Schiestel (1996). Numerical modelling of variable density turbulent jets. *International journal of heat and mass transfer* 39(9), 1865–1882.

Griebel, M. and P. Zaspel (2010). A multi-gpu accelerated solver for the three-dimensional two-phase incompressible navier-stokes equations. *Computer Science-Research and Development* 25(1-2), 65–73.

H ä llqvist, T. (2006). *Large eddy simulation of impinging jets with heat transfer*. Ph. D. thesis, Royal Institute of Technology.

Hirt, C., B. Nichols and N. Romero (1975). Sola: A numerical solution algorithm for transient fluid flows. Technical report, Los Alamos Scientific Lab., N. Mex. (USA).

Huai, Y. (2006). *Large Eddy Simulation in the scalar field*. Ph. D. thesis, Technische Universität.

Ilyushin, B. and D. Krasinsky (2006). Large eddy simulation of the turbulent round jet dynamics. Thermophysics and Aeromechanics 13(1), 43–54.

Jones, S., F. Sotiropoulos and M. Sale (2002). Large-eddy simulation of turbulent circular jet flows. Technical report, EERE Publication and Product Library, Washington, DC (United States).

Kumar, M. K., M. R. Abdel-Majeed and M. Annavaram (2019). Efficient automatic parallelization of a single gpu program for a multiple gpu system. *Integration* 66, 35–43.

Kuo, K. K. Y. and R. Acharya (2012). *Fundamentals of Turbulent and Multi-Phase Combustion*. John Wiley & Sons.

Lesieur, M., O. Métais and P. Comte (2005). *Large-eddy simulations of turbulence*. Cambridge University Press.

Lipari, G. and P. K. Stansby (2011). Review of experimental data on incompressible turbulent round jets. *Flow, turbulence and combustion* 87(1), 79–114.

Métais, O. and M. Lesieur (1992). Spectral large-eddy simulation of isotropic and stably stratified turbulence. *Journal of Fluid Mechanics* 239, 157–194.

Markesteijn, A. P., V. Semiletov and S. A. Karabasov (2015). Cabaret gpu solver for fast-turn-around flow and noise calculations. In *21st AIAA/CEAS Aeroacoustics Conference,* pp. 2223.

McMillan, O. (1980). Tests of new subgrid-scale models in strained turbulence. aiaa paper aiaa-80-1339. *In AIAA 13th Fluid and Plasma Dynamics Conference, Snowmass, Co.*

Moin, P., K. Squires, W. Cabot and S. Lee (1991). A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Physics of Fluids A: Fluid Dynamics* 3(11), 2746–2757.

Montorfano, A., F. Piscaglia and G. Ferrari (2013). Inlet boundary conditions for incompressible les: A comparative study. *Mathematical and Computer Modelling* 57(7-8), 1640–1647.

Morgans, R., B. Dally, G. Nathan, P. Lanspeary and D. Fletcher (1999). Application of the revised wilcox (1998) k-ω turbulence model to a jet in co-flow. *In Second International Conference on CFD in the Mineral and Process Industries, Melbourne, Australia.*

Pierce, C. D. (2001). *Progress-variable approach for large-eddy simulation of turbulent combustion*. Ph. D. thesis, stanford university.

Pierce, C. D. and P. Moin (2004). Progress-variable approach for large-eddy simulation of non-premixed turbulent combustion. *Journal of Fluid Mechanics* 504, 73–97.

Piomelli, U. (1999). Large-eddy simulation: achievements and challenges. *Progress in Aerospace Sciences* 35(4), 335–362.

Pope, S. B. and S. B. Pope (2000). *Turbulent flows. Cambridge university press.*

Quadros, A. E. R. (2016). *Processamento Paralelo em CUDA Aplicado ao Modelo de Geracão de cen´arios sint é ticos de vaz õ es e Energias-GEVAZP*. Ph. D. thesis, Universidade Federal do Rio de Janeiro.

Ruetsch, G. and M. Fatica (2011). Cuda fortran for scientists and engineers. *NVIDIA Corporation* 2701.

Sagaut, P. (2006). *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media.

Sagaut, P. (2013). *Multiscale and multiresolution approaches in turbulence: LES, DES and hybrid RANS/LES methods: applications and guidelines*. World Scientific.

Silva Freire, A. P., P. P. M. Menut and S. Jian (2002). *Turbulência, Volume* 1. ABCM.

Smagorinsky, J. (1963). General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review* 91(3), 99–164.

Tabor, G. R. and M. Baba-Ahmadi (2010). Inlet conditions for large eddy simulation: A review. *Computers & Fluids* 39(4), 553–567.

Tennekes, H. and J. L. Lumley (1972). *A first course in turbulence*. MIT press.

Thibault, J. and I. Senocak (2009). Cuda implementation of a navier-stokes solver on multi-gpu desktop platforms for incompressible flows. *In 47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, pp. 758.

Veynante, D. and L. Vervisch (2002). Turbulent combustion modeling. *Progress in Energy and Combustion Science* 28(3), 193–266.

Wang, P., J. Fröhlich, V. Michelassi and W. Rodi (2008). Large-eddy simulation of variable-density turbulent axisymmetric jets. *International Journal of Heat and Fluid Flow* 29(3), 654–664.

Wilcox, D. C. (2006). *Turbulence modeling for CFD. La Canada, CA: DCW Industries.* DCWI ndustries.

Wilson, R. V. and A. O. Demuren (1997). *Numerical simulation of turbulent jets with rectangular cross-section*. Number 97. National Aeronautics and Space Administration, Langley Research Center.

Wilson, T., B. Nichols, C. Hirt and L. Stein (1988). Sola-dm: A numerical solution algorithm for transient three-dimensional flows. Technical report, Los Alamos National Lab.

Zhu, X., E. Phillips, V. Spandan, J. Donners, G. Ruetsch, J. Romero, R. Ostilla-M ó nico, Y. Yang, D. Lohse, R. Verzicco, M. Fatica and R. J. A. M. Stevens (2018). Afid-gpu: a versatile navier–stokes solver for wall-bounded turbulent flows on gpu clusters. *Computer physics communications* 229, 199–210.