Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **3**, No. 3, pp. 370-388

DOI: 10.4208/aamm.10-10s2-07
June 2011

High Level Languages Implementation and Analysis of 3D Navier-Stokes Solvers

Valerio Grazioso[1,*], Carlo Scalo[1], Giuseppe de Felice[2] and Carlo Meola[2]

[1] *Department of Mechanical and Materials Engineering, Queen's University, 130 Stuart Street, Kingston, Ontario, Canada*
[2] *Dipartimento di Energetica Termofluidodinamica applicata e Condizionamenti ambientali (DETEC), Università degli Studi di Napoli 'Federico II', P.le Tecchio 80, 80125 Naples, Italy*

Received 29 January 2010; Accepted (in revised version) 28 October 2010
Available online 28 February 2011

Abstract. In this work we introduce PRIN-3D (PRoto-code for Internal flows modeled by Navier-Stokes equations in 3-Dimensions), a new high level algebraic language (Matlab®) based code, by discussing some fundamental aspects regarding its basic solving kernel and by describing the design of an innovative advection scheme. The main focus was on designing a memory and computationally efficient code that, due to the typical conciseness of the Matlab coding language, could allow for fast and effective implementation of new models or algorithms. Innovative numerical methods are discussed in the paper. The pressure equation is derived with a quasi-segregation technique leading to an iterative scheme obtained within the framework of a global preconditioning procedure. Different levels of parallelization are obtainable by exploiting special pressure variable ordering patterns that lead to a block-structured Poisson-like matrix. Moreover, the new advection scheme has the potential of a controllable artificial diffusivity. Preliminary results are shown including a fully three-dimensional internal laminar flow evolving in a relatively complex geometry and a 3D methane-air flame simulated with the aid of libraries based on the Flamelet model.

AMS subject classifications: 65M10, 78A48, 76M99.

Key words: Matlab, high level algebraical languages, segregation, preconditioning, flamelet.

1 Introduction

The vast majority of present CFD codes have been developed in what would be con-

*Corresponding author.
URL: `http://me.queensu.ca/people/piomelli/research/TSMLab.php`
Email: graziosov@me.queensu.ca (V. Grazioso), cscalo.ca@gmail.com (C. Scalo), giudefel@unina.it (G. de Felice), cmeola@unina.it (C. Meola)

sidered nowadays as low-level programming languages like C or Fortran. This allows for faster performances, and sometimes, for the creation of machine specific highly efficient executables. On the other hand, high-level or very high-level languages allow for higher level of abstraction from machine language. Important features include runtime interpretation and debugging, handling of more generic data structures, symbolic math toolboxes and intermediate code files. The well known drawbacks fall into the class of the so called abstraction penalties: slower execution speed, higher memory consumption, larger binary program size [4]. However, as computational resources get more sophisticated but yet more available and affordable, newly developed high-level programming languages become faster and easier to use and have the potential to be considered a valid candidate for code developing.

In order to design a CFD code that would enable the user to perform several tasks ranging from algebraic analysis of the equations' structure to modular implementation of virtually any kind of fluid dynamic model, it is therefore convenient to adopt high-level algebraic languages (like Python, Scilab, Octave, etc). For our purposes we considered Matlab ® as being the most appropriate option. This choice was made considering the opportunities that this kind of programming language offers in terms of easy user interfacing, fast libraries, packages integration (i.e., NAG, LAPACK, UMFPACK, etc) and increasing usage of natively multi-threaded functions (such as the fftw-based FFT). In comparison with low-level programming languages (like C or Fortran) many pre-compiled subroutines are available to the user and a very high-level of abstraction from the details of the computer is possible with, for instance, structures, cell-arrays and object oriented programming. The most important feature of Matlab remains its natural and native handling of fundamental linear algebra objects like matrices, and the extensive amount of libraries and functions available for a lot of simple and complex algebraic operations such as matrix multiplication, inversion or eigenvalues extraction. Moreover, Matlab has been developing parallelization capabilities (distributed and shared memory) by providing specifically designed toolboxes (MPI based) and packages (like Star-P®).

The choice of algebraic solving strategies is strongly dependant on the choice of the Navier-Stokes solver which is determined by the class of fluid dynamic problems one intends to solve. A first general distinction may be made among fully implicit, semi-implicit and fully explicit time discretization strategies. For steady laminar flows the transient evolution from an initial condition is usually not being resolved and large time-steps are used when possible; this requires methods that penalize time accuracy but that are, possibly, unconditionally stable in time allowing to reach steady state in the least amount of computational time. The best choice in this case would be SIMPLE-based solving techniques like PISO, SIMPLEC [2] in which both diffusive and (linearized) convective terms are treated implicitly allowing to remove CFL time constraints as well as viscous ones. For eddy resolving turbulent flow simulations the constraints on the time-step are determined by the need to accurately resolve the time scale of the smallest resolved eddies. The required time step is typically smaller than what determined by viscous and CFL constraints and, therefore, fully implicit schemes are a rather inconvenient choice for these type of simulations. Classic projection methods with semi-implicit time discretization and pseudo-pressure correction [6] typically represent the best choice in these cases.

Another crucially important step in the design of a CFD code is the choice of the convection scheme, which can also be problem-dependant. In laminar flows, where spatial grid requirements are not so demanding, large cell-size based Reynolds numbers may, however, cause stability problems. Great effort has been devoted in literature into designing stable but yet accurate convection schemes which could minimize the error due to the added artificial viscosity but at the same time accurately reconstruct the dynamics of advective transport of species and the velocity field itself. A large contribution comes from the research in compressible flow. In such context, due to the difficulties of the analysis for the multidimensional system, it is a common practice to test advection schemes and their properties in 1D. In the last four decades -after Godunovs theorem- the attempts to conjugate high order schemes with more advanced properties, like the stability and/or entropic ones, have been directed to less ambitious goals (e.g., monotonic, TVD, TVB, Muscl, ENO, WENO, etc), but more robust for the multidimensional extension [8]. The early Von Neumann ideas of using conservative schemes and adding a proper amount of artificial viscosity are still effectively implemented even if such numerical diffusion effects are introduced in a non explicit and non stationary way. Flux and slope limiters as well as solution adapted reconstruction are the main ingredients of most popular codes. For turbulent flows other constraints have to be met by convection scheme: large effort, for example, has been done design discrete energy conserving schemes [13] on non-uniform grids as well [16].

In this paper we describe the basic steps leading to the design of a 3D Navier-Stokes code. We decided to write from scratch a new numerical code called PRIN-3D (PRoto-code for Internal flows modeled by Navier-Stokes equations in 3-Dimensions)[†]. The purpose in mind was designing a flexible code (sometimes referred to as protocode) with turbulent and reactive capabilities and oriented to the study of new mathematical and numerical models and to the development or optimization of new numerical algorithms. The code is tailored to the general structure of the mathematical models that we want to solve (in particular the incompressible and the slightly compressible Navier-Stokes model) but also flexible enough to easily implement new models by exploiting the fast built-in functions present in Matlab. The focus is also on the design of advanced semi-implicit numerical solvers with pressure segregation by means of preconditioning techniques explained in Section 2, and the design of a memory and computational efficient version of it described in Section 2.1. In Section 2.2 we describe the new advection scheme in which we combine a Lagrangian prediction (based on a solution adapted reconstruction) for the set up of low and high order Richtmyer-Lax-Wendroff fluxes obtaining a new scheme, successfully checked for both 1D scalar case and 3D Navier-Stokes equations. In Section 3 we show the results from two simula-

[†]The code is opensource and downloadable at `http://sourceforge.net/projects/prin-3d/`.

tions. The first one is a laminar flow with fully three-dimensional boundary conditions meant to show the flexibility of the code. The second is a combustion simulation, fully three-dimensional as well, which shows good agreement with experimental data.

2 Solving kernel

In this section we are presenting the structure of PRIN-3D's linear solver for the system of equations, resulting from the discretization of the unsteady incompressible Navier-Stokes partial differential equations. This represents the basic solving kernel for every incompressible or even slightly compressible model being resolved. An implicit time discretization for the diffusive part, weighing with θ the unknowns at time $n+1$ and with $(1-\theta)$ the known values at time n, is adopted (the most common choice for the value of θ is $1/2$ and this leads to the Cranck-Nicolson scheme with a truncation error $e = \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$). All the non linear terms are treated explicitly. The pressure gradient time collocation is intended at the time $t^{(n+1)}$ to stress the fact that the vector field $\partial P^{n+1}/\partial x_j$ has to correct the divergence of the finally computed field at time $t^{(n+1)}$. In incompressible flows pressure has no thermodynamical meaning and its collocation in time has no physical meaning.

Introducing a spatial discretization as well, the whole problem-in a Cartesian Orthogonal reference system- can be restated in terms of the following algebraic operators

$$
\begin{aligned}
&F\, v_j^{n+1} + G_j\, p^{\,n+1} = q_j, \quad j = 1,2,3,\\
&D_i\, v_i^{\,n+1} = g.
\end{aligned}
$$

Pressure nodes p and velocity nodes v_j are located respectively at the centre and at the vertices of cubic control volumes (partially staggered variable colocation). The computational grid is a 3D block-structured grid obtained by a staircase approximation of the user-defined boundary geometry and the mesh is uniform in all directions with $\Delta x = \Delta y = \Delta z$. The partially-staggered option appears to be a fair trade off between fully collocated and fully staggered variable arrangement. The symbology can be further more condensed in the following manner

$$
\begin{pmatrix} \mathcal{F} & \mathcal{G} \\ \mathcal{D} & 0 \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} q \\ g \end{pmatrix}, \tag{2.1}
$$

where $\mathcal{F}$ is a block-diagonal elliptic operator $\left(\begin{smallmatrix} F & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & F \end{smallmatrix}\right)$, $\mathcal{G}$ is a column operator $\left(\begin{smallmatrix} G_1 \\ G_2 \\ G_3 \end{smallmatrix}\right)$, $\mathcal{D}$ is a row operator $\left(\begin{smallmatrix} D_1 & D_2 & D_3 \end{smallmatrix}\right)$, while

$$
v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \quad \text{and} \quad q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}.
$$

In order to solve the system (2.1), a possible choice is to consider a classic Richardson iteration scheme with the following left preconditioner

$$\mathcal{P} = \begin{pmatrix} \mathcal{F} & 0 \\ \mathcal{D} & -I \end{pmatrix}, \tag{2.2}$$

that can be interpreted in terms of a Chorin-like projection method. In fact, for every iterative step

$$\begin{pmatrix} \mathcal{F} & 0 \\ \mathcal{D} & -I \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,\ iter+1} = - \begin{pmatrix} 0 & -\mathcal{G} \\ 0 & I \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,\ iter} + \begin{pmatrix} q \\ g \end{pmatrix}, \tag{2.3}$$

an intermediate velocity field is computed (momentum equations are solved with no the pressure-gradient term on the left-hand side) and then the new pressure field is, with virtually no cost, obtained from the computation of the divergence of such field. Supposing such method to be convergent, the final solution will satisfy the following preconditioned system (see [1,5,10])

$$\begin{pmatrix} \mathcal{F}^{-1} & 0 \\ \mathcal{D}\mathcal{F}^{-1} & -I \end{pmatrix} \begin{pmatrix} \mathcal{F} & \mathcal{G} \\ \mathcal{D} & 0 \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} q' \\ g' \end{pmatrix}, \tag{2.4}$$

that with simple algebraic manipulation can be rewritten as

$$\begin{pmatrix} I & \mathcal{F}^{-1}\mathcal{G} \\ 0 & \mathcal{D}\mathcal{F}^{-1}\mathcal{G} \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} \mathcal{F}^{-1}q \\ \mathcal{D}\mathcal{F}^{-1}q - g \end{pmatrix}. \tag{2.5}$$

This is the typical structure of a segregated system where the coupling between the velocity and pressure variables in the computations has been weakened (i.e., the operator becomes block triangular). As a matter of fact, the segregation procedure can also be suggested by the usage of the Theorem on the Inverse Problem of Vector Field Calculus [9]. This gives a fair interpretation of the pressure field as a scalar function that provides the divergence correction for the velocity field[‡] and, thus, has to satisfy the classic pressure-segregation or Poisson's equation

$$\mathcal{D}\mathcal{F}^{-1}\mathcal{G}p^{n+1} = \mathcal{D}\mathcal{F}^{-1}q - g. \tag{2.6}$$

This process involves the computation of the inverse $\mathcal{F}^{-1}$ or of the group $\mathcal{F}^{-1}\mathcal{G}$. Since the coefficient matrix of the linear system of Eqs. (2.1) remains unchanged throughout the whole computation (advection term is estimated with explicit techniques such as predictor-corrector), such inverse could be precomputed and stored in memory. This option is clearly not recommendable since if the matrices $\mathcal{F}^{-1}$ and $\mathcal{F}^{-1}\mathcal{G}$ were directly computed, both would turn out to be completely full and their storage would be dramatically demanding on memory resources. No sparse memory storage is therefore possible. However, only by solving exactly (2.6), in which $\mathcal{D}\mathcal{F}^{-1}\mathcal{G}$ is completely full as well, we can enforce the divergence-free constraint on the updated velocity field.

[‡]This interpretation is exact when using a fully explicit time advancement.

2.1 Sparse approximate inverse: quasi-segregated discrete equations

As previously stated, performing the full segregation process, by means of direct solving, is too expensive especially in the 3D case (see [3,7]). A smarter choice, therefore, would be to adopt the following series expansions of the inverse of F

$$F^{-1} = \big(I - \nu\theta\Delta t\ \widehat{\nabla}^2\big)^{-1} = I + \sum_{n=1}^{\infty} \big(\nu\theta\Delta t\ \widehat{\nabla}^2\big)^n,$$

where $\widehat{\nabla}$ is the discrete Laplace operator applied to velocity variables. The same expression is valid for the matrix blocks present in the diagonal of the inverse of $\mathcal{F}$. Such approximation converges only if

$$\nu\theta\Delta t\rho(\widehat{\nabla}^2) < 1,$$

where $\rho(\widehat{\nabla}^2)$ indicates the spectral radius of $\widehat{\nabla}^2$. The series can be truncated at the N-th order thus obtaining the following approximate inverse operator

$$\widetilde{F}_N^{-1} = I + (\nu\theta\Delta t\ \widehat{\nabla}^2) + (\nu\theta\Delta t\ \widehat{\nabla}^2)^2 + \cdots + (\nu\theta\Delta t\ \widehat{\nabla}^2)^N, \tag{2.7}$$

with an obvious reduction of the sparsity for larger values of N. Instead of using the full inverse of F, the approximate N-th order inverse (2.7) is adopted for each of the three diagonal blocks of $\mathcal{F}^{-1}$, yielding the following sparse approximate block diagonal matrix

$$\widetilde{\mathcal{F}}_N^{-1} = \begin{pmatrix} \widetilde{F}_N^{-1} & 0 & 0 \\ 0 & \widetilde{F}_N^{-1} & 0 \\ 0 & 0 & \widetilde{F}_N^{-1} \end{pmatrix}. \tag{2.8}$$

Before proceeding any further, it is important to (numerically) analyze and possibly control the error due to the adoption of the sparse approximate inverse form (2.7) instead of the full inverse one,

$$F^{-1} = \Big[I - (\nu\theta\Delta t\ \widehat{\nabla}^2)\Big]^{-1}.$$

Let D be an n-by-n matrix so that D/h^2 is the numerical approximation on uniform mesh, with

$$\Delta x = \Delta y = \Delta z = h,$$

of the discrete Laplacian $\widehat{\nabla}$ operator with homogeneous Dirichlet boundary conditions. The operator F will then be $(I - \beta D)$ with

$$\beta = \frac{\nu\theta\Delta t}{h^2}.$$

The error of a first order (i.e., $N = 1$) sparse approximate inverse defined as

$$\frac{||(I + \beta\widehat{\nabla}^2) - (I - \beta\widehat{\nabla}^2)^{-1}||}{||(I - \beta\widehat{\nabla}^2)^{-1}||},$$

increases with the matrix size but it is an almost quadratic function of β for large values of n. To control such error, a value of $\beta_{\max} = 0.02$ has been chosen. For a given mesh size, viscosity and θ, it will be an upper bound for the time step interval Δt, apart from other constraints given, for example, by the CFL condition. Moreover, such a small value of $\beta_{\max}$ was needed in order to contain losses in accuracy.

The segregation process (2.4) is now repeated by using the approximate inverse (2.8) instead of the actual $\mathcal{F}^{-1}$ Bearing in mind that

$$\widetilde{F}_N^{-1} F = I - B_N = I - (\nu\theta\Delta t\ \widehat{\nabla}^2)^{N+1}, \tag{2.9}$$

this yields

$$\begin{pmatrix} I - \mathcal{B}_N & \widetilde{\mathcal{F}}_N^{-1}\mathcal{G} \\ -\mathcal{D}\mathcal{B}_N & \mathcal{D}\widetilde{\mathcal{F}}_N^{-1}\mathcal{G} \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} \widetilde{\mathcal{F}}_N^{-1} q \\ \mathcal{D}\widetilde{\mathcal{F}}_N^{-1} q - g \end{pmatrix}. \tag{2.10}$$

The pressure variables in this system are quasi-segregated (not fully segregated as in the (2.5) system) since $\mathcal{B}_N$ is different from zero but of the $(N+1)^{\text{th}}$ order in Δt (for fixed ν and θ). Every term with $\mathcal{B}_N$ can be either neglected or brought to the right-hand side, therefore starting the following iterative method which will be denoted with the i iterative counter:

$$\begin{pmatrix} I & \widetilde{\mathcal{F}}_N^{-1}\mathcal{G} \\ 0 & \mathcal{D}\widetilde{\mathcal{F}}_N^{-1}\mathcal{G} \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,i+1} = \begin{pmatrix} \mathcal{B}_N & 0 \\ \mathcal{D}\mathcal{B}_N & 0 \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,i} + \begin{pmatrix} \widetilde{\mathcal{F}}_N^{-1} q \\ \mathcal{D}\widetilde{\mathcal{F}}_N^{-1} q - g \end{pmatrix}. \tag{2.11}$$

The system of Eqs. (2.11) is clearly block-triangular and when solving for the $v^{n+1,i+1}$ unknowns, the following pressure-segregation equation has to be solved first

$$\mathcal{D}\widetilde{\mathcal{F}}_N^{-1}\mathcal{G} p^{n+1,i+1} = \mathcal{D}\mathcal{B}_N v^{n+1,i} + \mathcal{D}\widetilde{\mathcal{F}}_N^{-1} q - g. \tag{2.12}$$

Moreover, it is possible to split each approximate inverse operator $\widetilde{F}_N^{-1}$ into two parts $I + F_1$ and F_2, as follows

$$\widetilde{F}_N^{-1} = I + \sum_{n=1}^{N_{lhs}} (\nu\theta\Delta t\ \widehat{\nabla}^2)^n + \sum_{n=N_{lhs}+1}^{N} (\nu\theta\Delta t\ \widehat{\nabla}^2)^n = I + F_1 + F_2.$$

With such splitting, Eq. (2.12) is ready to be solved with an iterative method. The order of the $I + F_1$ term to be left on the left-hand side of the equation is determined by N_{lhs}, and this leads to the following "nested" iterative scheme indicated with the counter k:

$$\mathcal{D}(I + \mathcal{F}_1)\mathcal{G} p^{n+1,i+1,k+1} = -\mathcal{D}\mathcal{F}_2\mathcal{G} p^{n+1,i+1,k} + \mathcal{D}\mathcal{B}_N v^{n+1,i} + \mathcal{D}\widetilde{\mathcal{F}}_N^{-1} q - g. \tag{2.13}$$

If $N_{lhs} = 0$ (i.e., $\mathcal{F}_1 = 0$), the Eq. (2.13) has the same coefficient matrix of the pressure equation in an explicit time discretization. The higher the value of N_{lhs} the less sparse will the coefficient matrix $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ be. Choosing $N_{lhs} \geq 1$ gives a bi-harmonic nature to such equation [15], reducing the pressure checkerboard effect, while, on the

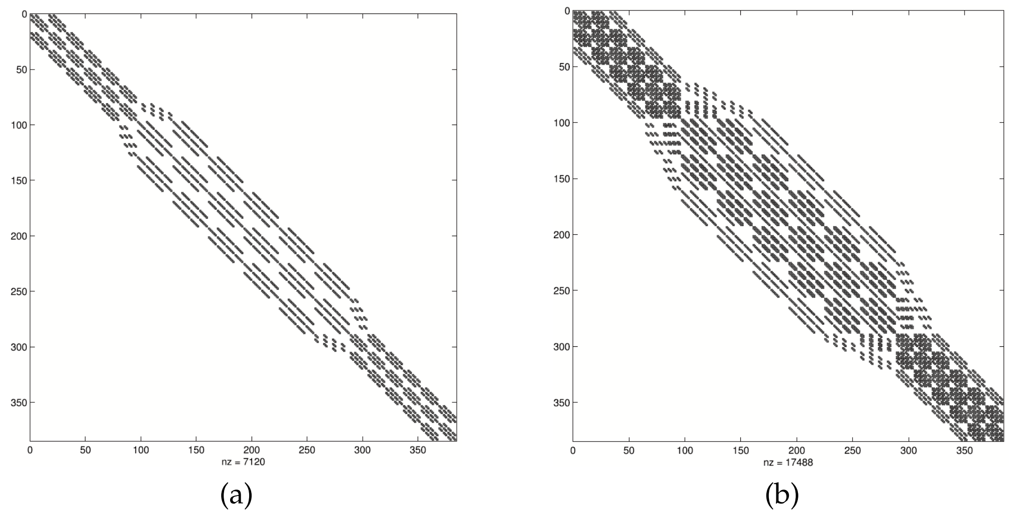


Figure 1: Matrix pattern visualization of $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=0$ (a) and with $N_{lhs}=1$ (b).

other hand, increasing the memory and computational demands for solving such system (see Figs. 1(a) and (b)).

The Galerkin matrix $\left(\begin{smallmatrix}\mathcal{F} & \mathcal{G}\\ \mathcal{D} & 0\end{smallmatrix}\right)$ to be solved is illustrated in Fig. 3. In this case a sequential variable ordering (shown in Fig. 2) has been adopted and, for expositive purposes, a coarse mesh (384 pressure nodes) is used.

Performing the previously described solving procedure with such variable ordering, the pattern of the final pressure equation's coefficient matrix for $N_{lhs}=0$ and $N_{lhs}=1$ (with $N\geq 1$) will respectively be as in Figs. 1(a) and (b).

Apparently there is no further possible nested iterative cycle that could be exploited to solve Eq. (2.13). Let's analyze the stencil of an internal equation given by the matrix $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ (that, for the sake of simplicity will be chosen with $N_{lhs}=0$) which is shown in Fig. 4. The idea is to look for alternative pressure variable orderings so that the elliptic operator $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ exhibits a block-structured pattern. This will make it possible to introduce another nested iterative cycle on Eq. (2.13) apart from allowing

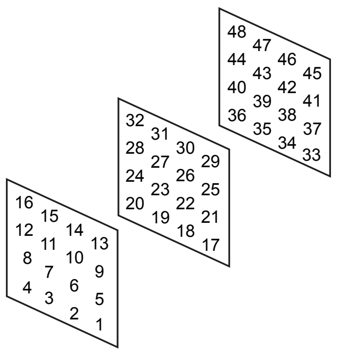


Figure 2: Lexicographic variable ordering.

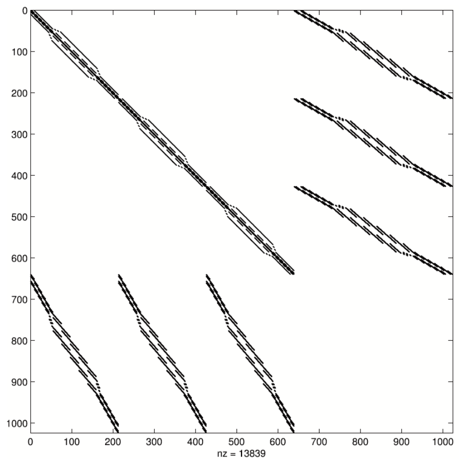


Figure 3: Matrix pattern visualization of the Galerkin matrix corresponding to lexicographic ordering.

a parallel resolution for such equation. The alternative pressure variable ordering that are being examined are the following:

1. Two pressure "colors" (classic 3D red-black ordering), Fig. 5(a), yielding patterns as shown in Figs. 7(a) and (b),
2. Three pressure "colors", Fig. 5(b), yielding patterns as shown in Figs. 8(a) and (b),
3. Four pressure "colors", Fig. 6, yielding patterns as shown in Figs. 9(a) and (b).

All of the pressure orderings from 2 to 4 make it possible to set up a further iterative nested cycle (with counter l). Two strategies have been pursued: a block Gauss-Seidel or SOR method for a non-parallel solving technique and a block Gauss-Jacobi method which allows parallel computing on clusters or on multi-threaded single machine.

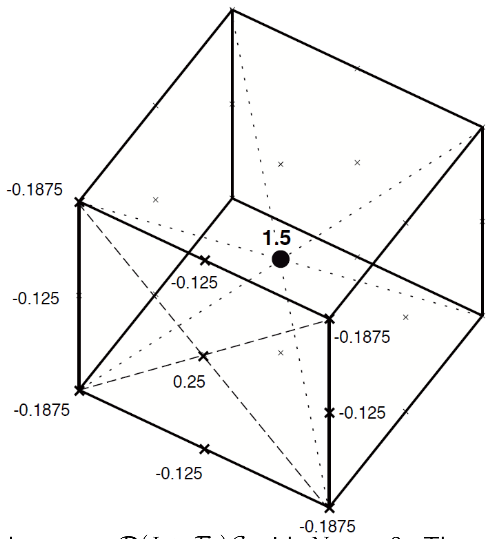
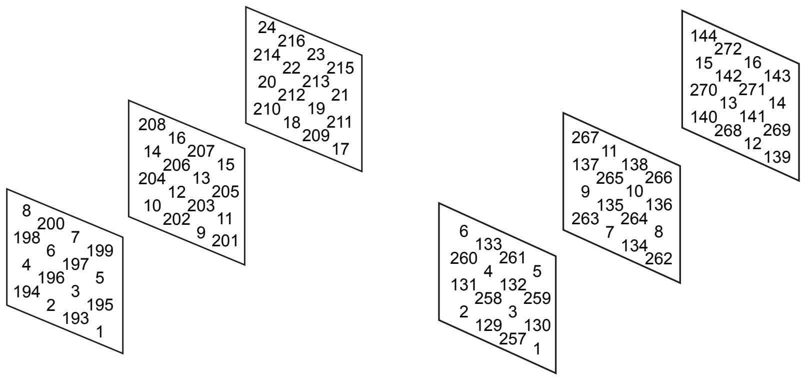


Figure 4: Stencil of numerical operator $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=0$. The weights are repeated identically for every cube face (isotropic elliptic operator), the central weight is $+1.5$.



(a) (b)

Figure 5: Two colors pressure variable ordering (red-black) (a); Three colors pressure variable ordering (b).

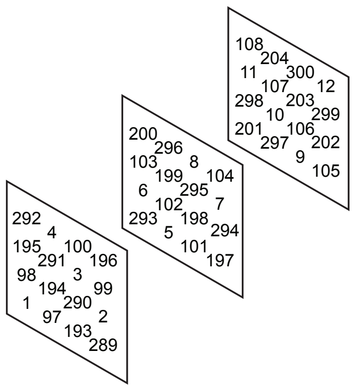
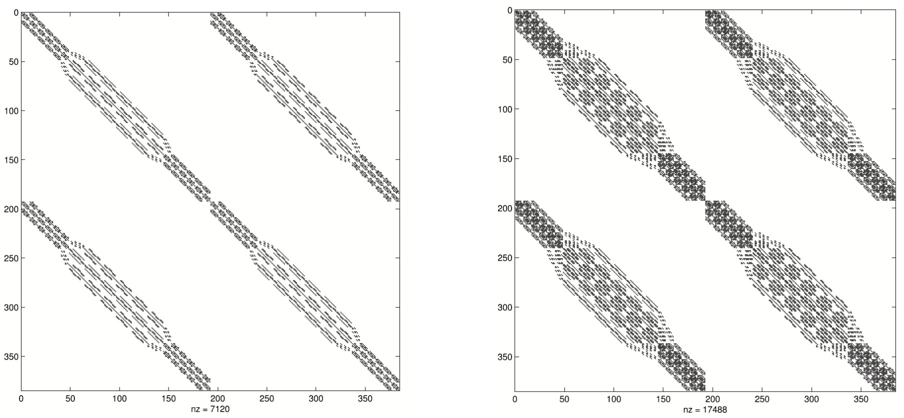
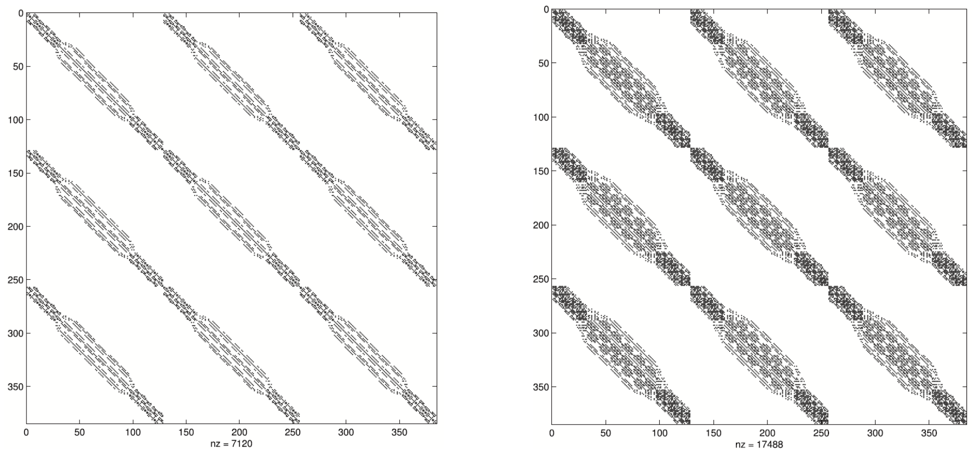


Figure 6: Four colors pressure variable ordering.



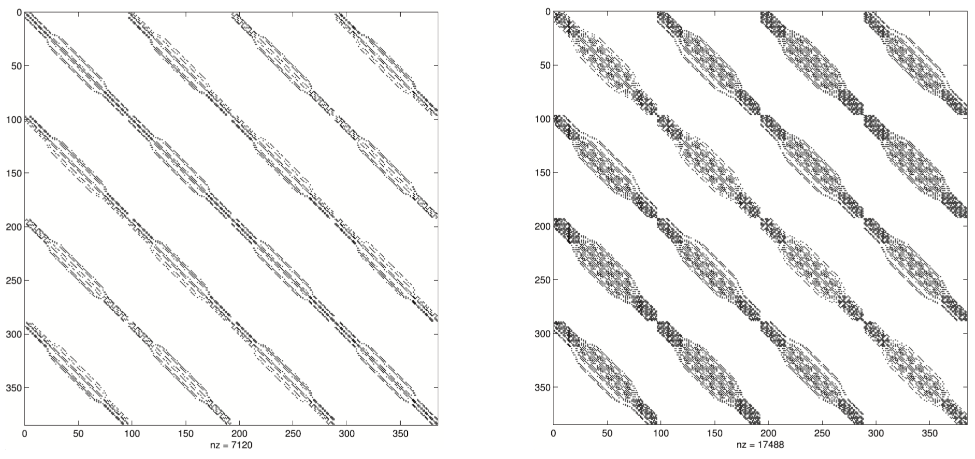
(a) (b)

Figure 7: Pattern visualization (two colours) of $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=0$ (a), with $N_{lhs}=1$ (b).



(a) (b)

Figure 8: Pattern visualization (three colours) of $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=0$ (a), with $N_{lhs}=1$ (b).



(a) (b)

Figure 9: Pattern visualization (four colours) of $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=0$ (a), with $N_{lhs}=1$ (b).

2.2 Treatment of non-linear terms: multilevel Lagrangian conservative scheme

The non-linear terms have been treated in a fully explicit way, as already mentioned. Many advection schemes are available in the code among which an innovative and completely original scheme, the Multilevel Lagrangian Conservative Scheme (MLCS) that is certainly worth discussing in its essential aspects (more details can be found in [17]). For clarity purposes we will refer to a simple 1D advection problem in the present section. The MLCS scheme is based on the Richtmyer scheme but with the possibility to control the artificial viscosity present in the scheme. The first step consists in interpolating with a given order the nodal values of the transported component of momentum u from the first grid (grid 1) to a second, finer, grid (grid 2) (indicated

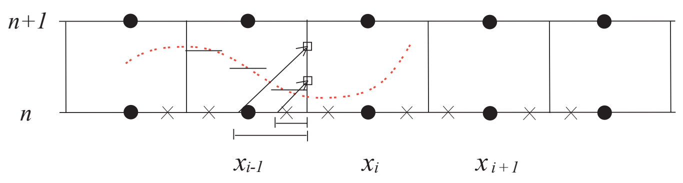


Figure 10: Illustration of the MLCS steps.

respectively with full dots and crosses in Fig. 10. Once the values of u are available on the finer grid we can estimate the time integral of the flux of u on the cell faces. Let's consider the time integration of the flux on the face between the cell i and $i-1$. Such integral is carried out by means of Gaussian integration in time of the analytical flux $f(u)$ at this location leading to the following expression for the time averaged flux

$$\tilde{f} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(u_{i-\frac{1}{2}})dt \simeq \frac{1}{2}\Big[f(u_{i-\frac{1}{2}}^{n+\frac{3-\sqrt{3}}{6}}) + f(u_{i-\frac{1}{2}}^{n+\frac{3+\sqrt{3}}{6}})\Big] = \tilde{F}. \tag{2.14}$$

This approximation is third order accurate in time. The values of the velocity at the Gauss points in time are calculated as follows:

$$u_{i-\frac{1}{2}}^{n+\frac{3\pm\sqrt{3}}{6}} = u^n\left(x_{i-\frac{1}{2}} - \frac{3\pm\sqrt{3}}{6}\Delta t\ u^*_{i-\frac{1}{2}}\right),$$

with

$$u^*_{i-\frac{1}{2}} = \frac{1}{2}(u_{i-1} + u_i).$$

The values of

$$u^n\Big(x_{i-\frac{1}{2}} - \frac{3\pm\sqrt{3}}{6}\ \Delta t\ u^*_{i-\frac{1}{2}}\Big),$$

are obtained with a nearest value interpolation from the array of interpolated velocities on the grid 2, providing the correct amount of artificial viscosity while preserving a high-order spatial and temporal resolution. By increasing the order of such interpolation we can reduce the magnitude of the added artificial viscosity.

3 Results

Three-dimensional geometry handling is one of the toughest steps in a CFD code design. An effort has been made to guarantee the maximum of flexibility with respect to the user's demands in within a not too limited class of 3D domains. The actual version of the code is intended to handle a specific class of three-dimensional domains, right prisms. The user defines a basic 2D polygon and the height of the prism which extends over the third dimension.

As described above the computational grid is a 3D block-structured grid obtained by a staircase approximation of the user defined boundary geometry and the mesh is evenly spaced ($\Delta x = \Delta y = \Delta z$).

Boundary Conditions (BC) available for the velocity's components u, v, w are Dirichlet (for all faces) and Extrapolation (for Outflow faces only) (Pressure Inlet and Neumann BC are supported in the code but not yet completely tested). It is also possible to assign lid driven faces or even swirled ones. Dirichlet and Neumann BC are available for transported scalars and other model-related quantities such as Turbulent Energy K, Energy Dissipation rate ε and Mixture Fraction Z.

In this section we show the results from two simulations: a laminar flow with fully three-dimensional boundary conditions meant to show the flexibility of the code; fully three-dimensional methane-air flame, which shows good agreement with experimental data.

3.1 Fully 3D

The boundary conditions consist of two Dirichlet outflows with circular plug flow velocity profiles, one with negative V component the other with W positive component, and one inflow where a negative U component has been assigned as shown in Fig. 11. The code has automatically checked and readjusted such velocity boundary conditions in order to satisfy the divergence-free constraint and this was an important test for the right-hand side equation generators because the inflow and outflow conditions are totally three-dimensional. Since this was a simple test with no physical relevance, a non conservative Lagrangian convective scheme (with 3D linear interpolation with low computational cost) has been adopted. Such type of convective scheme has proven to be less numerically dissipative than a TVD high order flux reconstruction with dimensional splitting. Predictor-Corrector convective schemes with Lagrangian prediction are also available in the code and could be a better option for accuracy improvement.

The simulation data are shown in Table 1 where Lx, Ly, Lz are the extensions of the domain in the x, y, z direction, Np_{tot} and Nv are respectively the total numbers of pressure cells and velocity nodes and Re is the Reynolds number based on the highest magnitude boundary velocity and the inflow orifice diameter.

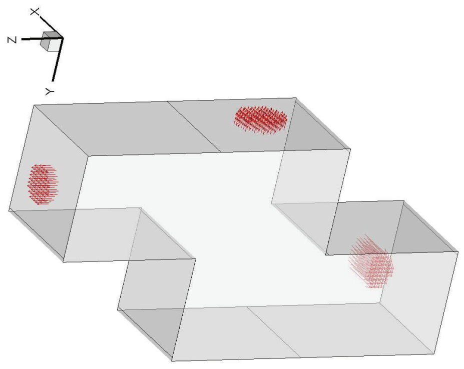


Figure 11: Rappresentation of dirichlet velocity boundary conditions.

Table 1: General simulation data for Fully3D.

$Lx = 0.5$	$Nv = 110862$
$Ly = 1$	$Re = 31.54$
$Lz = 2$	$Np_{tot} = 119700$

This simulation has been carried out with the second type of pressure variable ordering because during the testing phase the fourth pressure variable ordering and block Gauss-Seidel (or SOR) solving were causing oscillations in the pressure field. Similar effects were found in the flow field, increasing the possibility of arising numerical instabilities. It is possible to notice from Fig. 12 that the pressure iso-surfaces' values indicate that it decreases downstream (as it should be).

It is interesting to zoom into the inflow area Fig. 13 and notice that there is a stagnation point on the wall right in front of the inlet, and that right on the edge of the inflow orifice there are recirculating streamlines (see Fig. 14).

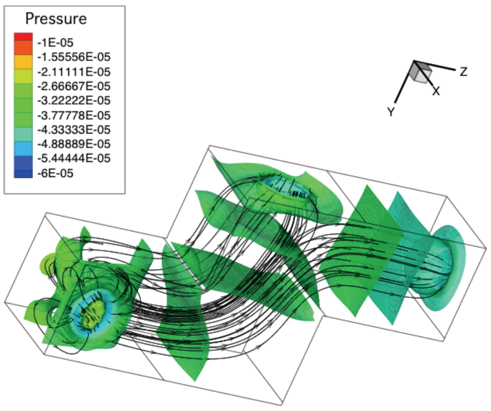


Figure 12: Post-processing of a fully 3D Incompressible Navier Stokes simulation.

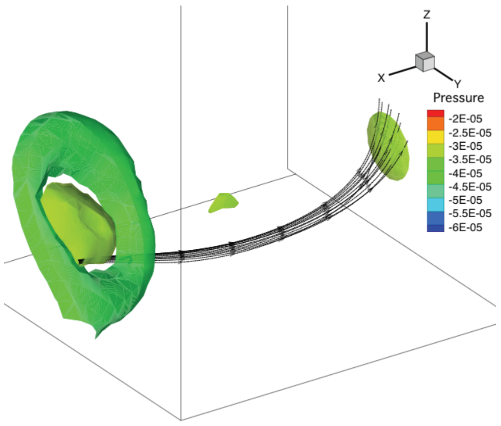


Figure 13: Stagnation point zoom in.

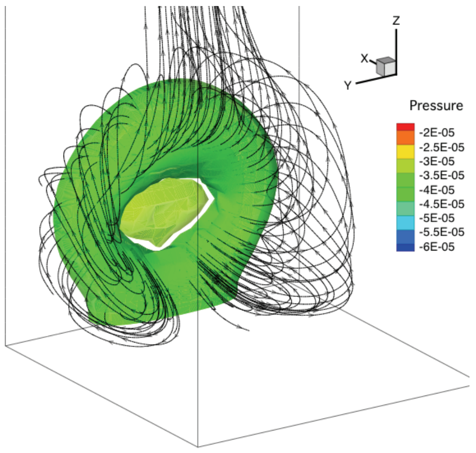


Figure 14: Inflow recirculating streamlines.

3.2 SANTORO 3D flame

The results of a 3D combustion simulation done by adopting the 3D boundary conditions of the experiment in [14] are shown here. The computational domain is a 3D box with a single inflow and outflow face. Axial-symmetric boundary conditions (as far as the staircase approximation's limited reconstruction capability is concerned) are reproduced by using combinations of circular hat functions. On the same inflow face, starting from its centre, radial functions for the mixture fraction, normal velocity component, density and dynamic viscosity are assigned, in order to reproduce the geometry of the coannular burner and the concentric air annulus inflows in [14].

The combustion model used here is based on the Flamelet model [11], [12] which accounts for the effects on the velocity field of the local interaction between oxidizer and fuel, by means of mixture fraction transport, from which the scalar dissipation rate (proportional to the mixture fraction gradient magnitude) is computed. From these quantities all thermochemical conditions at every point can be determined.

The simulation parameters are shown in Table 2. Np_x, Np_y, Np_z, Nv are, respectively, the number of pressure nodes in the x, y, z direction and the total number of velocity nodes while Lx, Ly, Lz define the extent of the computational domain in the x, y, z direction (in meters).

Table 2: 3D Flame simulation data.

$Np_y = 67$	$Nv = 392040$	$Np_{tot} = 200715$	$Np_x = 67$
$Np_z = 91$	$Lx = 0.11$	$Ly = 0.11$	$Lz = 0.15$

3.2.1 Plug inflow conditions

As a first simulation attempt, for both fuel and oxidizer's velocity inflow conditions, plug profiles have been adopted. Moreover, because of the large number of pressure cells and of the bulk geometry that leads to a quite filled pseudo-elliptic matrix pattern, a first attempt was made with the four colour pressure variable arrangement (see Section 2.1 which is the fastest but least stable).

Because of these choices, as it can be seen in Fig. 15, the flame, in its maximum extension, is shorter than expected and in addition it collapsed due to the known stability issues associated with the four pressure variable arrangement.

As it can be observed from Figs. 16(a) and (b), the effect of combustion on the flow is to generate mass flux source cells. This due to the fact that in the reacting cells temperature rises causing density to drop and particles to be locally pushed away from the reacting front. In particular, from Fig. 16(b) it is clear how such positive-divergence front, where non stoichiometric reactions are taking place, stays ahead of the actual flame surface.

Table 3: 3D Flame simulation data-Plug Flow.

$Re_{FUEL} = 63.59$	$Re_{FUELcell} = 9.56$	$Re_{OX} = 109.94$	$Re_{OXcell} = 16.53$

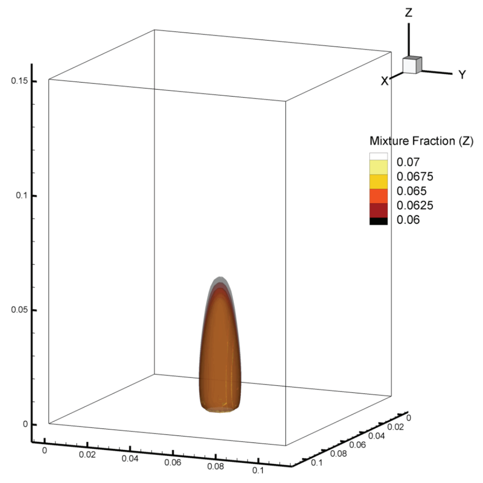
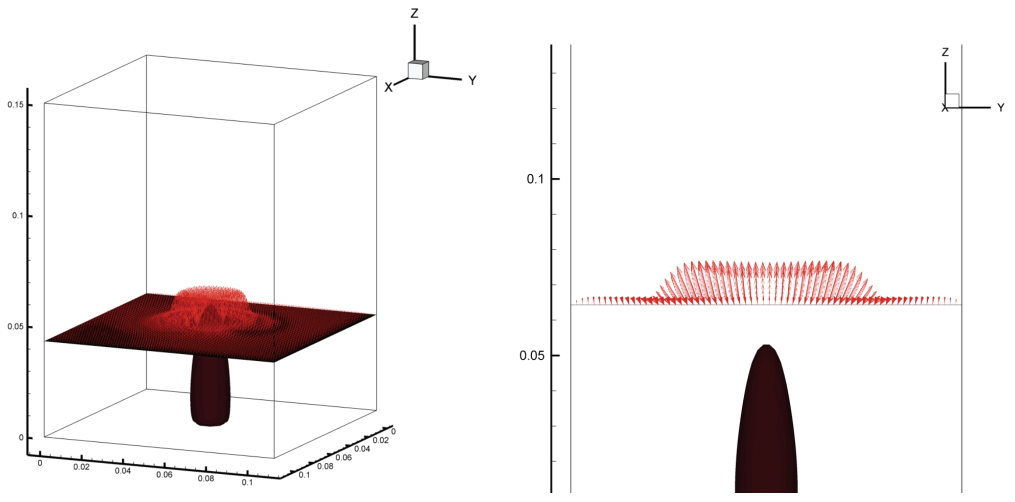


Figure 15: Santoro 3D flame Z profile.



(a) (b)

Figure 16: Santoro 3D flame. Mass flux source cells (a). Santoro 3D flame. Chemical reaction front (b).

3.2.2 Poiseuille inflow conditions

Observing that in [14] is specified that fuel and oxidizer supply ducts are long enough to obtain fully developed flows, it was decided to impose, for both fuel and oxidizer inlet velocities, Poiseuille flow conditions as shown in Fig. 17.

In terms of radial coordinate (r) such inlet conditions are:

$$w_{FUEL} = 1 - \frac{r^2}{R_1^2},$$

for the fuel, and

$$w_{AIR} = R_2^2 - r^2 + (R_2^2 - R_1^2)\frac{\ln(R_2/r)}{\ln(R_1/R_2)},$$

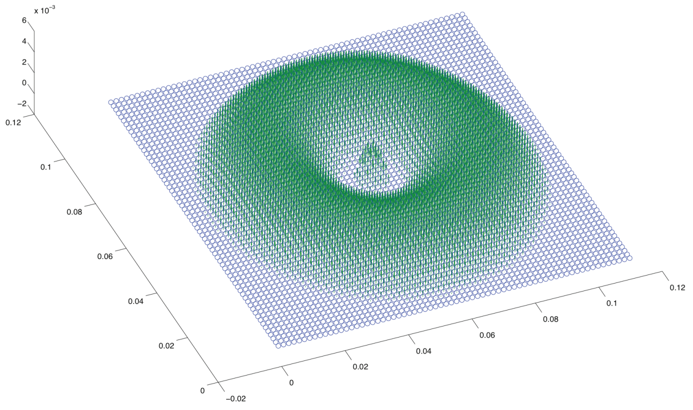


Figure 17: Santoro 3D flame. Poiseuille inflow conditions.

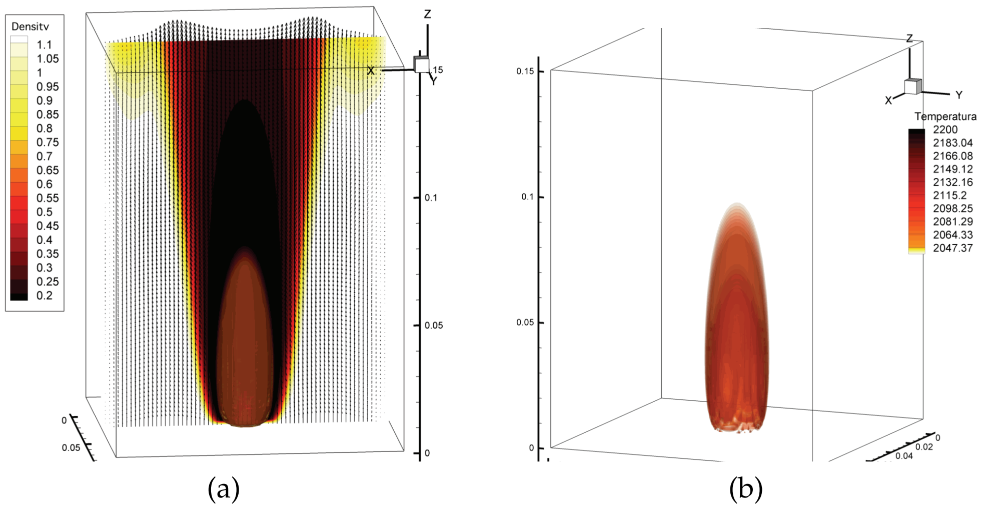


Figure 18: Santoro 3D flame. Density profile and velocity vectors slice (a). Temperature iso-surfaces (b).

for the oxidizer, where R_1 and R_2 are respectively the inner and outer radius of the air annulus.

For the same net flow rate of the previous case, such profiles have higher maximum velocities than the plug ones and a longer flame is obtained (as expected). A slice of the velocity vector field and the density distribution is shown Fig. 18(a). We can notice the higher value of the density of the surrounding air and its gradual transition to the low value of methane inside the flame, through the flame surface. In Fig. 18(b) temperature iso-surfaces close to the maximum value are shown and they provide a good indication of the size of the flame and of the stoichiometric front.

The temperature distribution at $Z = 0.0762$m is shown in Fig. 19(b) together with

Table 4: 3D Flame simulation data.

$Re_{FUEL} = 127.18$	$Re_{FUELcell} = 19.12$	$Re_{OX} = 164.91$	$Re_{OXcell} = 24.80$

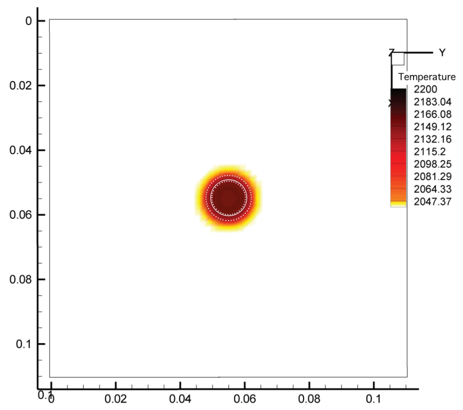


Figure 19: Santoro 3D flame. Temperature distribution at $Z = 0.0762$m plane. Experimental position of the stoichiometric value of the mixture fraction (in between two dashed circles), isoline of stoichiometric value of mixture fraction predicted by PRIN-3D (solid circle).

the available experimental data at this location [14]. The dashed circles define the circular annulus in which the experimental position of the stoichiometric value of the mixture fraction falls. The solid circle is the isoline of the mixture fraction predicted by our simulation at the same value.

4 Conclusions

PRIN-3D is a high level algebraical proto-code designed to assist in the study of old and new mathematical and numerical models and to the development or optimization of new numerical algorithms. Its numerical solving kernel is based on the quasi-segregation of pressure elliptic equation and subsequent resolution by means of nested iterative processes with parallel capabilities. A wide variety of convective numerical schemes are also implemented for the treatment of the non-linear terms of the N. S. Equations ranging from simple -non resource demanding- Lagrangian to an original, more sophisticated Multilevel Lagrangian Conservative Scheme just recently developed [17].

Some preliminary, but non trivial, numerical tests show good agreement with available experimental data.

References

[1] S. Badia and R. Codina, *Algebraic pressure segregation methods for the incompressible naveri-stokes equations*, Arch. Comput. Methods. Eng., 15 (2008), pp. 343–369.

[2] J. H. Ferziger and M. Peric, Computational Methods for Fluid Dynamics, Springer-Verlag, Berlin, 2002.

[3] A. George and J. W. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, New Jersey, 1981.

[4] A. CHATZIGEORGIOU AND G. STEPHANIDES, *Evaluating performance and power of object-oriented vs. procedural programming languages*, Proceedings of the 7th Ada-Europe International Conference on Reliable Software Technologies, Springer.

[5] V. E. HOWLE, J. SHADID, H. ELMAN AND R. TUMINARO, *A parallel block multi-level preconditioner for the 3d incompressible Navier-Stokes equations*, J. Comput. Phys., 187 (2003), pp. 504–523.

[6] J. KIM AND P. MOIN, *Application of a fractional-step method to incompressible Navier-Stokes equations*, J. Comput. Phys., 59 (1985), pp. 308–323.

[7] A. ERISMAN, I. DUFF AND J. REID, Direct Methods for Sparse Matrices, Oxford University Press, Oxford, 1986.

[8] R. J. LEVEQUE, Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, New York, 2002.

[9] C. MEOLA AND G. DE FELICE, *Fondamenti lineari per la Fluidodinamica Numerica*, Edizioni l'Ateneo, Napoli, 1996.

[10] R. PEYRET, Handbook of Computational Fluid Mechanics, Academic Press, London, 2000.

[11] H. PITSCH, *Creating a flamelet library for the steady flamelet model or the flamelet/progress variable approach*, User manual of: FlameMaster, A C++ Computer Program for 0D Combustion and 1D Laminar Flame Calculations, September 2006.

[12] H. PITSCH AND N. PETERS, *A consistent flamelet formulation for non-premixed combustion considering differential diffusion effects*, Combust. Flames., 114 (1998), pp. 26–40.

[13] Y. MORINISHI, T. S. LUND, O. V. VASILYEV AND P. MOIN, *Fully conservative higher order finite difference schemes for incompressible flow*, J. Comput. Phys., 143 (1998), pp. 90–124.

[14] R. J. SANTORO, R. PURI AND K. C. SMYTH, *The oxidation of soot and carbon monoxide in hydrocarbon diffusion flames*, Combust. Flames., 97 (1994), pp. 125–144.

[15] C. M. RHIE AND W. L. CHOW, *Numerical study of the turbulent flow past an airfoil with trailing edge separation*, AIAA J., 21 (1983), pp. 1525–1532.

[16] F. E. HAM, F. S. LIEN, AND A. B. STRONG, *A fully conservative second-order finite difference scheme for incompressible flow on nonuniform grids*, J. Comput. Phys., 177 (2002), pp. 117–133.

[17] V. GRAZIOSO, N. MASSAROTTI, C. MEOLA AND C. SCALO, *A multilevel Lagrangian conservative scheme (extended abstract)*, First International Conference on Computational Methods for Thermal Problems, pp. 183–186, Naples September 8-10, 2009.