Comparisons of weakly-compressible and truly incompressible approaches for viscous flow into a high-order Cartesian-grid finite volume framework

L. Vittoz, G. Oger, M. de Leffe, D. Le Touzé

Version of Record: https://www.sciencedirect.com/science/article/pii/S2590055219300319
Manuscript_0e8b52a37708ce8b5a2e81bd895fed96

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Comparisons of weakly-compressible and truly incompressible approaches for viscous flow into a high-order Cartesian-grid finite volume framework

L. Vittoz[a,b], G. Oger[a,*], M. de Leffe[b], D. Le Touzé[a]

[a]*Ecole Centrale Nantes, LHEEA Research dept. (ECN and CNRS), Nantes, France*
[b]*Nextflow Software, Nantes, France*

ARTICLE INFO

Article history:

Keywords:
Incompressible flows
Pressure Poisson equation
Weakly-compressible approach
Locally refined mesh
Cartesian grid
High-order finite volume

ABSTRACT

An original strategy to address hydrodynamic flow was recently proposed through a high-order weakly-compressible Cartesian grid approach [1]. The method, named Weakly-Compressible Cartesian hydrodynamics (WCCH), is based on a fully-explicit temporal scheme for solving the Navier-Stokes equations while implicit incompressible schemes are usually preferred in the literature to address such flows. The present study aims to position and compare the WCCH method with a standard incompressible formulation. To this end, an incompressible scheme has been implemented in the same numerical framework. As far as possible, the algorithm used in the incompressible approach has been designed to be the same as (or close to) the one used in the weakly-compressible approach. In particular, high-order schemes for spatial and time discretization are employed. Pros and cons for each formulation are discussed in conjunction with a series of test cases on extensive criteria including implementation convenience, easy use of mesh refinement, convergence order and accuracy, numerical diffusion, parallel CPU scaling for high performance computing, etc. These comparisons demonstrate the relevance of the incompressible approach, at least for the selected test cases.

1. Introduction

While any fluid is compressible in reality, most liquid flow display negligible density variations and the fluid can be considered as incompressible. Following this hypothesis, such flows can be addressed in three different ways: (i) with a truly compressible, (ii) a weakly-compressible or (iii) an incompressible solver (see Fig. 1 for the energy

*Corresponding author
e-mail: guillaume.oger@ec-nantes.fr (G. Oger)

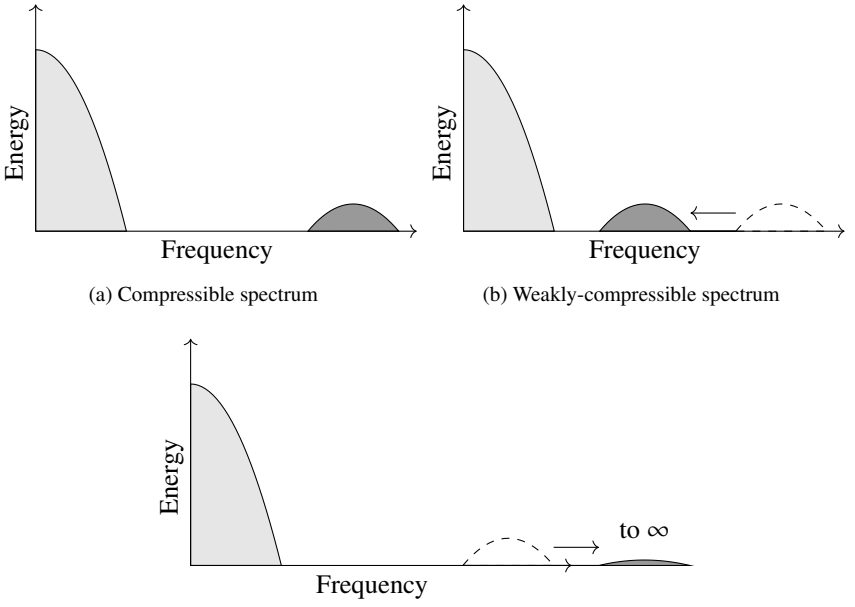
spectrum corresponding to each approach), depending on how is treated the acoustic part of the flow (in dark grey in Fig. 1)

Truly compressible approaches are often ill-adapted to quasi-incompressible hydrodynamics (corresponding to low Mach number flows). Indeed, in this particular case, compressible solvers are known to show a relatively poor accuracy and to require some very small time steps which are inappropriate with respect to the low frequency evolution of the incompressible part of the flow (plotted in light grey in Fig. 1a). To overcome this problem, two techniques are commonly used; asymptotic preserving schemes and preconditioning methods as proposed by Turkel [2] and Guillart et al. [3]

The weakly-compressible approach consists in keeping the governing equations of the compressible approach and to select an arbitrary value for the speed of sound c so that the numerical scheme accepts a larger time step without compromising the near-incompressibility assumption. Indeed, by using an explicit scheme for time integration, a stability constraint on the Courant number should be respected. Increasing the value of c drastically decreases the time step. In practice, modelling a hydrodynamic flow with the actual speed of sound leads to very low Mach number values. For instance, the speed of sound is equal to $c = 1480 m/s$ in water under standard conditions for temperature and pressure. Compressible effects can be considered as negligible provided that the Mach number of the flow $Ma = \frac{u}{c}$ stays lower than 0.1. Therefore, c can be decreased to increase the time step as long as the Mach number is lower than 0.1. The loss of accuracy is negligible because the energy part related to the compressible part is not significant as illustrated in Fig. 1b. The weakly-compressible approach can be very convenient since it leads to a straightforward implementation. Pressure can be obtained explicitly from an algebraic thermodynamic equation without solving any linear system. For instance, weakly-compressible schemes are commonly used with the smoothed particle hydrodynamics (SPH) method [4]. Explicit algorithms can efficiently run on distributed memory architectures with an arbitrary high number of computational cores while maintaining good scaling properties (see [5] for the SPH method). These observations led us to develop the WCCH method to address hydrodynamic flow. This approach as well as the main features of the code (high order schemes for both spatial and time discretization, Cartesian grid with local mesh refinement capabilities, immersed boundary method, etc.) are described in [1].

In the truly incompressible approach, the compressibility tends towards zero so that acoustic perturbations are instantaneously propagated throughout the domain, which is equivalent to a sound speed tending to infinity (Fig. 1c). In the (weakly-)compressible strategy, the fluid is treated in a more realistic way since all fluids (gas or liquid) are actually compressible. Nevertheless, the propagation of information at the speed of sound can result in drastic CFL conditions for high Reynolds number flows unless a fully implicit scheme is used, while incompressible resolutions can alleviate this limitation. In the incompressible approach, the constant density hypothesis is equivalent to a velocity divergence-free constraint. As a result, the pressure should be obtained implicitly after solving the linear system of the Pressure Poisson Equation. In the past decades, many schemes have been suggested to solve the incompressible Navier-Stokes equation with Finite Volume method. Firstly, a projection approach was separately proposed by Chorin and Temam [6, 7] and also from Harlow and Welch [8]. Later on pressure-based methods like SIMPLE [9, 10], PISO [11], etc. were intensively developed, essentially to address steady flows. In the context of AMR for Cartesian grids

and the immersed boundary method, the projection scheme using an incompressible approach is the most commonly used method [12, 13]. This scheme is flexible and remains relatively easy to code and use in comparison with the SIMPLE method which implicitly solves the momentum and Pressure Poisson Equations. In the projection approach, the convective and diffusive parts of the momentum equations can be solved explicitly. Thus, it is highly compatible with high-order schemes for the discretization of these terms. Furthermore, only the pressure should be solved implicitly so that this technique can be characterised as semi-implicit. Among all the numerical schemes applied to incompressible approaches, the basic Chorin's projection scheme has been selected while maintaining the efficiency and accuracy of the WCCH solver.



(a) Compressible spectrum

(b) Weakly-compressible spectrum

(c) Incompressible spectrum

Fig. 1: Differences between compressible, weakly-compressible and incompressible approaches. ▫ : purely incompressible part, ■ : acoustic part.

This paper is dedicated to presenting an alternative incompressible approach to the WCCH method which conserves its capabilities, and to comparing the performance of both schemes in terms of accuracy and efficiency. The governing equations are introduced first in Section 2. Then the weakly-compressible formulation and its implementation (already described in [1]) are recalled in Section 3 to emphasise the modification needed in the numerical framework to include an incompressible scheme (Section 4). Several test cases are presented to validate the incompressible implementation and to compare it with the weakly-compressible solver (Section 5). Finally, the high performance computing features and parallelization efficiency of both approaches are discussed in Section 6. Comparisons between the weakly-compressible and incompressible schemes have already been conducted using the SPH method [14, 15, 16]. Artificial compressibility methods [17, 18] have also been compared with incompressible schemes [19, 20]. However, weakly-compressible and artificial compressibility methods are rather different especially for unsteady cases. To the best of our knowledge, there is no equivalent comparison between weakly-compressible and

incompressible approaches in the Finite Volume framework, especially in presence of high-order schemes. This paper addresses this specific point.

2. Governing equations

The Navier-Stokes equations describing the motion of the fluid can be expressed as:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \\ \dfrac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \nabla \cdot (\overline{\overline{\boldsymbol{\tau}}}) + \rho \boldsymbol{f} \end{cases} \tag{1}$$

where ρ, $\boldsymbol{u}$, p, $\overline{\overline{\boldsymbol{\tau}}}$ and $\boldsymbol{f}$ denote respectively the density, the velocity vector, the pressure, the viscous stress tensor and an external force such as gravity or body forces for instance. The viscous stress tensor is related to the velocity field through a constitutive law, which is defined for compressible Newtonian fluids as:

$$\overline{\overline{\boldsymbol{\tau}}} = 2\mu \left[\frac{\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T}{2} - \frac{1}{3} (\nabla \cdot \boldsymbol{u}) \overline{\overline{\boldsymbol{I}}} \right] \tag{2}$$

where μ is the dynamic viscosity and $\overline{\overline{\boldsymbol{I}}}$ is the identity tensor.

3. Weakly-compressible formulation and numerical schemes

3.1. Equation of state

In the solver of the WCCH method, the system of Eq. (1) can be closed, for instance, with the following barotropic equation of state:

$$p = \frac{\rho_0 c_0^2}{\gamma} \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right) \tag{3}$$

where γ is the polytropic constant of the fluid, ρ_0 its nominal density and c_0 its nominal speed of sound. For example, γ is commonly set to 1.4 for air and 7 for water. Since this relation between p and ρ does not involve the fluid energy, the conservation of energy does not need to be solved.

3.2. Numerical schemes

The system of Eq. (1) can be summarised as:

$$\frac{\partial \boldsymbol{W}}{\partial t} + \nabla \cdot \Psi_c(\boldsymbol{W}) = \nabla \cdot \Psi_v(\boldsymbol{W}) + \boldsymbol{S} \tag{4}$$

with the convective variables vector $\boldsymbol{W} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \end{pmatrix}$,

the convective part $\nabla \cdot \Psi_c(\boldsymbol{W}) = \underbrace{\begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \end{pmatrix}_{,x}}_{\boldsymbol{F}_c(\boldsymbol{W})_{,x}} + \underbrace{\begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \end{pmatrix}_{,y}}_{\boldsymbol{G}_c(\boldsymbol{W})_{,y}} + \underbrace{\begin{pmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \end{pmatrix}_{,z}}_{\boldsymbol{H}_c(\boldsymbol{W})_{,z}}$,

the viscous fluxes $\nabla \cdot \Psi_v(\boldsymbol{W}) = \underbrace{\begin{pmatrix} 0 \\ \mu\tau_{xx} \\ \mu\tau_{yx} \\ \mu\tau_{zx} \end{pmatrix}_{,x}}_{\boldsymbol{F_v}(\boldsymbol{W})_{,x}} + \underbrace{\begin{pmatrix} 0 \\ \mu\tau_{xy} \\ \mu\tau_{yy} \\ \mu\tau_{zy} \end{pmatrix}_{,y}}_{\boldsymbol{G_v}(\boldsymbol{W})_{,y}} + \underbrace{\begin{pmatrix} 0 \\ \mu\tau_{xz} \\ \mu\tau_{yz} \\ \mu\tau_{zz} \end{pmatrix}_{,z}}_{\boldsymbol{H_v}(\boldsymbol{W})_{,z}}$ and the source term $\boldsymbol{S} = \begin{pmatrix} 0 \\ \rho \boldsymbol{f} \end{pmatrix}$. The indices $_{,x}$, $_{,y}$ and $_{,z}$ refer to the partial derivative with respect to x, y and z respectively. The viscous stress tensor is defined as:

$$\overline{\overline{\tau}} = \begin{pmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{pmatrix} \tag{5}$$

In this work, a Finite Volume method is used to discretize the conservation Eq. (4) with collocated variables. They are discretized in space and time separately according to the method of lines. After integrating these equations onto a Cartesian grid and applying the divergence theorem, the following semi-discrete equations are obtained

$$\begin{aligned} \frac{d\boldsymbol{W}_{i,j,k}}{dt} = & \frac{1}{\Delta x}(\boldsymbol{F}_{i-\frac{1}{2},j,k} - \boldsymbol{F}_{i+\frac{1}{2},j,k}) + \\ & \frac{1}{\Delta y}(\boldsymbol{G}_{i,j-\frac{1}{2},k} - \boldsymbol{G}_{i,j+\frac{1}{2},k}) + \\ & \frac{1}{\Delta z}(\boldsymbol{H}_{i,j,k-\frac{1}{2}} - \boldsymbol{H}_{i,j,k+\frac{1}{2}}) + \\ & \frac{1}{\Delta x \Delta y \Delta z}\boldsymbol{S}_{i,j,k} \end{aligned} \tag{6}$$

with $\boldsymbol{W}_{i,j,k} = \frac{1}{\Delta x \Delta y \Delta z} \iiint_{C_{i,j,k}} \boldsymbol{W}(x,y,z)dxdydz$ representing the average value of the conservative variables in cell $C_{i,j,k} = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}] \times [z_{k-\frac{1}{2}}, z_{k+\frac{1}{2}}]$ where $\boldsymbol{F} = \boldsymbol{F}_c - \boldsymbol{F}_v$, $\boldsymbol{G} = \boldsymbol{G}_c - \boldsymbol{G}_v$, $\boldsymbol{H} = \boldsymbol{H}_c - \boldsymbol{H}_v$. These latter fluxes are evaluated at the midpoint of the cells faces. The convective flux requires to reconstruct all variables at these locations. This is achieved using a 5^{th} order WENO scheme [21] and the VFFC solver proposed by Ghidaglia et al. [22] is used to evaluate the numerical flux. For the viscous part, the stress tensor is calculated with a space-centered second order velocity gradient at the cells face.

Finally, time integration is performed using a 4^{th} order Runge-Kutta scheme. Indeed , the 5th order spatial scheme of the convective part requires a high order time discretization scheme to maintain the overall convergence order of the solver. The time step is defined according to the following CFL condition to ensure the stability of this explicit temporal scheme:

$$\Delta t = \min\left(C_c \min_{i\in\{x,y,z\}} \left(\frac{\Delta i}{\boldsymbol{u}\cdot\boldsymbol{n}_i + c} \right), C_v \min_{i\in\{x,y,z\}} \left(\rho \frac{\Delta i^2}{\mu} \right) \right) \tag{7}$$

where C_c and C_v are the Courant and diffusion numbers, Δ the cell size and $\boldsymbol{n}_i$ the normal vector in direction $i \in \{x,y,z\}$. In addition, the fourth order of accuracy of the time integration scheme allows to set C_c up to 0.7 and $C_v = 0.125$ [1]. For example, the vortex convection test case described in section 5 can be run with a Courant number set to 0.7 while an explicit 1^{st} order Euler method would require to lower this number to 0.05. For high Reynolds number flows, the time step is also limited by the convective part.

4. Incompressible approach

The incompressible approach assumes a constant density, so that the system of equations (1) becomes

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 \\ \dfrac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\dfrac{\nabla p}{\rho} + \dfrac{1}{\rho} \nabla \cdot (\bar{\bar{\boldsymbol{\tau}}}) + \boldsymbol{f} \end{cases} \tag{8}$$

In comparison to the weakly-compressible approach, pressure and density are decoupled and pressure appears exclusively in the momentum equation through a gradient operator. A possible solution to overcome this difficulty is to calculate pressure so that the continuity equation is satisfied through the Pressure Poisson Equation. Note that the term $-\frac{1}{3}(\nabla \cdot \boldsymbol{u})\bar{\bar{\boldsymbol{I}}}$ from the viscous stress tensor formula (2) which corresponds to the bulk viscosity is strictly null in the incompressible approach. The viscous part can be simplified as :

$$\frac{1}{\rho} \nabla \cdot (\bar{\bar{\boldsymbol{\tau}}}) = \nu \Delta \boldsymbol{u} \tag{9}$$

where ν is the kinematic viscosity.

4.1. Projection scheme

The projection scheme was proposed separately by Chorin and Temam [6, 7]. It consists of a first prediction of the velocity through an explicit discretization of the convective and diffusive parts of the momentum equation. This predicted velocity is then corrected with the pressure gradient determined after solving the Pressure Poisson Equation. The correction step ensures that the corrected velocity is divergence-free.

1. Prediction step

$$\boldsymbol{u}^* = \boldsymbol{u}^n + \Delta t (\nu \Delta \boldsymbol{u}^n - \nabla \cdot (\boldsymbol{u}^n \otimes \boldsymbol{u}^n) + \boldsymbol{f}) \tag{10}$$

2. Pressure Poisson Equation

$$\nabla \cdot \left(\frac{\Delta t}{\rho} \nabla p^{n+1} \right) = \nabla \cdot \boldsymbol{u}^* \tag{11}$$

3. Correction step

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \frac{\Delta t}{\rho} \nabla p^{n+1} \tag{12}$$

The time integration scheme used here is the same 4$^{\text{th}}$ Runge-Kutta method used in the weakly-compressible approach. Applying explicit Runge-Kutta schemes to the incompressible Navier-Stokes equations is not straightforward due to the differential-algebraic nature of the equations and the specific role of the pressure which is necessarily implicit. Note that the use of high-order time integration schemes combined with Chorin's projection is seldom used in literature, where time accuracy is often limited to the second-order. For instance, the fractional step method by Kim and Moin [23] is second order accurate in time through the Adams-Bashforth scheme. Le and Moin suggested a fractional step method based on a third order Runge-Kutta scheme [24]. Nevertheless, the use of a fourth order Runge-Kutta scheme in projection schemes has been recently proposed in the literature [25, 26]. The commonly

utilised technique is to apply the projection scheme at each sub-step of the RK scheme. The pressure is solved at each sub-step to ensure the divergence-free constraint of the intermediate velocity:

$$\boldsymbol{u}^{*,i} = \boldsymbol{u}^n + \Delta t \sum_{j=1}^{i} \tilde{a}_{ij} \boldsymbol{RHS}^j \ , \ i = 1, .., 4 \tag{13}$$

$$\nabla \cdot \frac{\tilde{c}_i \Delta t}{\rho} \nabla p^i = \nabla \cdot \boldsymbol{u}^{*,i} \tag{14}$$

$$\boldsymbol{u}^i = \boldsymbol{u}^{*,i} - \frac{\tilde{c}_i \Delta t}{\rho} \nabla p^i \tag{15}$$

RHS denotes the discretization of the convective and viscous parts of the Navier-Stokes equations, in addition of the source terms. $\tilde{a}_{ij}$ and $\tilde{c}$ are coefficients of the explicit RK4 method written in compact form, and using the Butcher's array :

$$\tilde{A} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{pmatrix}, \tilde{c} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \\ 1 \end{pmatrix} \tag{16}$$

This way, the velocity might be fourth-order accurate in time whereas the temporal convergence order of the pressure obtained at the last sub-step remains the same as the original projection scheme (i.e. only first-order accurate in the L_∞-norm, as demonstrated by Brown et al. [27] and Guermond et al. [28]). Indeed, an artificial boundary layer due to the Dirichlet boundary condition on the velocity limits the overall accuracy of the Chorin's projection. Many schemes had been proposed to overcome this difficulty (see for instance Kim and Moin [23] who implicitly treat the viscous part and use an incremental projection scheme).

Recently Sanderse et al. [29] analyzed the temporal order of accuracy of the velocity and pressure and proposed the Runge-Kutta method to compute a "post-treated pressure" at the best possible convergence-order. For a classical RK4 scheme, they showed that the pressure can be calculated using a combination of the pressure obtained at each sub-step:

$$p^{n+1} = -p^1 - 2p^2 - 2p^3 + 6p^4 \tag{17}$$

This gives a fourth-order pressure in time with periodic boundary conditions, but remains first-order accurate with Dirichlet boundary conditions applied to the velocity. This high-order pressure does not change the projection scheme. In this work, thereafter, all pressure values mentioned as "post-treated pressure" are calculated using (17). Otherwise, the pressure is derived from the last sub-step.

The incompressible approach was implemented in the same numerical framework so that both formulations can be compared. Thus, the numerical schemes are the same as those described in Section 3. The non-linear convective term $\nabla \cdot (\boldsymbol{u}^n \otimes \boldsymbol{u}^n)$ of Equation (10) is still discretized using the 5^{th} order WENO scheme from Shu [21]. The viscous part of the Navier-Stokes equations is calculated with a second-order centred scheme as well. Furthermore, incompressible implementation has been briefly introduced and already validated for several test cases in [30].

4.2. Location of variables

In the projection scheme, the location of variables has a major impact on the solution since the use of collocated variables leads to a local grid decoupling (checkerboard instabilities). Almgren et al. [31, 32] introduced an*approximate projection* concept that can deal with collocated variables; the divergence-free constraint is imposed on the reconstructed face-centred velocities rather than of the cell-centred velocities. It can be robust and is able to deal with high Reynolds number. For instance, the solvers *Gerris* [12] and *Basilisk* [33] successfully integrate the *approximate projection* within an octree-AMR framework. *Approximate projection* is usually based on the second-order upwind method proposed by Bell, Colella, and Glaz (also referred as the BCG scheme) [34]. However, this method is limited to a second-order for both spatial and time discretization accuracy. Recently, new high-order schemes for a collocated grid based on spectral deferred correction methods have been suggested by Minion et al. [35, 36] and Almgren et al. [37]. However, these methods require a large number of projection steps and have been only validated for uniform Cartesian mesh. Alternatively, a staggered grid arrangement does not suffer from this limitation of the *approximate projection* and is inherently stable. Therefore, in the present work we preferred the use of a staggered grid arrangement as in the Marker-And-Cell (MAC) scheme [8], in order to be compatible with high-order spatial schemes where the velocity variables are stored on cells faces and the pressure variables are cell-centred (see Fig. 2). In return, interlacing the unknown location generates three dual grids in 2D (four in 3D) and exacerbates the

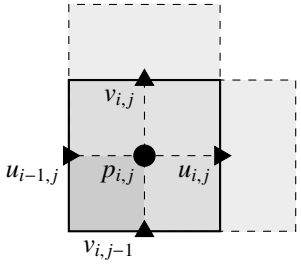


Fig. 2: Unknown variable locations

complexity of the memory storage and allocation.

4.3. Pressure Poisson Equation

The divergence and gradient operators involved in the Pressure Poisson Equation ((11)) are discretized with space-centred second-order schemes. The divergence operator is applied to the cell face unknowns, like velocity or pressure gradient, and is evaluated at cell centre (Fig. 3a).

$$\nabla \cdot \boldsymbol{u} = \frac{u_{i,j} - u_{i-1,j}}{\Delta x} + \frac{v_{i,j} - v_{i,j-1}}{\Delta y} \tag{18}$$

On the other hand, the gradient operator is attached to cell centre unknowns (pressure) for a cell face evaluation (Fig. 3b).

$$\frac{\partial p}{\partial x} = \frac{p_{i+1,j} - p_{i,j}}{\Delta x} \tag{19}$$

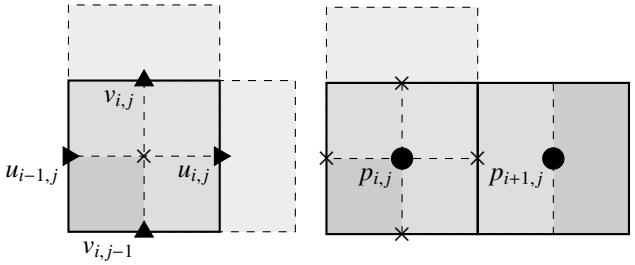


Fig. 3: Discretization stencil for divergence (left) and gradient operators (right) at the location ×

Regarging the resolution of the linear system, geometric multigrid methods are known to be efficient for solving elliptic problems such as the Pressure Poisson Equation. These fast linear iterative solvers use a hierarchy of grids of varying level of discretization covering the entire computation domain. However, in a first attempt, we chose to solve this linear system through the Krylov solver restarted GMRES, from PETSc library [38], in combination with the algebraic multigrid (AMG) from the HYPRE preconditioners library [39]. When Dirichlet boundary conditions are applied to the velocity, the linear system related to the Pressure Poisson Equation is singular since pressure is only determined up to a constant value. To address this problem, an additional constraint is required, for example:

$$\iiint_D p(x,y,z,t)dxdydz = 0 \tag{20}$$

Another solution to this problem, is to set the pressure of a given cell to zero by replacing the row of the Laplacian matrix by zero entries except for the diagonal entry which is set to 1. For this cell, the right-hand-side term of the Poisson Equation is set to a desired pressure value.

4.4. Local grid refinement

The local grid refinement represents an interesting technique to limit computational costs while maintaining high accuracy. An octree-based mesh refinement strategy is used in the present work (see for instance [40]). A hierarchy of block cover the entire computation domain. The blocks form the node of the octree with a fixed number of cells per block. Note that this article is mainly devoted to the comparison between weakly-compressible and incompressible schemes, so that the description of the discretization scheme on the locally refined grid is not detailed here. Briefly, the computation of the convective and diffusive parts at the refinement interfaces is performed by a local modification of the interpolation stencils to preserve the high-order accuracy and the stability as much as possible. The pressure Poisson Equation is also modified in order to ensure mass conservation near refinement interfaces. Finally, the linear system is still solved using the algebraic multigrid.

4.5. Immersed boundary method

To account for flows past complex geometries within a Cartesian grid, the immersed boundary method proposed in [1] is extended to the present incompressible solver. This method is a first order direct-forcing approach in which a source term is added to the Navier-Stokes equations to prescribe the body velocity onto the cells located inside

the solid as described by Fadlun et al. [41]. Since the exact position of the boundaries is not taken into account (through interpolated velocities for example), the source term is gradually enforced for solid cells with the regularized Heaviside step function

$$H(d) = \frac{1}{2}(1 + \tanh(-Kd)) \tag{21}$$

where d is the signed distance between the current cell centre and the body (taken as positive outside the solid and negative inside). K is a coefficient chosen so as to adjust the bandwidth of the transition region between the fluid and the solid. $H = 1$ inside the body while $H = 0$ far from the boundary in the fluid. The regularization procedure is similar to the solid fraction volume weighting proposed by Fadlun et al. [41] or Kajishima et al. [42]. Despite the relative simplicity of this method, accurate results were obtained on laminar flow, as shown in [1].

We propose to adapt this method to the present an incompressible scheme by applying the following:

$$\begin{cases} \boldsymbol{u}^* = \boldsymbol{u}^n + \Delta t\left[\left(1 - H\right)\left(\nu\Delta\boldsymbol{u}^n - \nabla\cdot(\boldsymbol{u}^n \otimes \boldsymbol{u}^n) - \dfrac{\nabla p^n}{\rho}\right) + H\boldsymbol{f}\right] \\ \nabla\cdot\left(\dfrac{\Delta t}{\rho}\nabla\psi^{n+1}\right) = \nabla\cdot\boldsymbol{u}^* \\ \boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \dfrac{\Delta t}{\rho}\nabla\psi^{n+1} \text{ and } p^{n+1} = p^n + \psi^{n+1} \end{cases} \tag{22}$$

An incremental version of the projection scheme is used here as proposed in [43]. A pressure gradient value ∇p^n is added to the prediction step. The pressure increment ψ is solved with the Poisson equation. Following [41], the source term is obtained with $\boldsymbol{f} = \frac{\boldsymbol{u}_{body} - \boldsymbol{u}^n}{\Delta t}$. Furthermore, the regularized source term $H\boldsymbol{f}$ represents the force applied to the body. In return, the introduction of the source term limits the overall accuracy in time to the first order, in particular for the velocity field. This effect is not so significant for the pressure field which is computed anyway at a low order accuracy (as outlined previously). We found this incremental version to be more suitable for the immersed boundary method, since the predicted velocity is corrected with a low magnitude term (in comparison with the standard scheme) and because the source term actually prescribes the predicted velocity $\boldsymbol{u}^*$ rather than the final velocity $\boldsymbol{u}^{n+1}$. Besides, for steady flow, the pressure increment ψ tends to 0, allowing the enforcement of both the divergence-free constraint and the kinematic boundary condition. Note that the predicted velocity $\boldsymbol{u}^*$ can be reformulated as:

$$\boldsymbol{u}^* = (1 - H)\boldsymbol{u}^n_{fluid} + H\boldsymbol{u}_{body} \tag{23}$$

to explicitly decompose it into a fluid and solid velocity contribution. H represents an equivalent to the solid volume fraction [41, 42].

$$\boldsymbol{u}^n_{fluid} = \boldsymbol{u}^n + \Delta t\left(\nu\Delta\boldsymbol{u}^n - \nabla\cdot(\boldsymbol{u}^n \otimes \boldsymbol{u}^n) - \frac{\nabla p^n}{\rho}\right) = \boldsymbol{u}^n + \Delta t\boldsymbol{RHS}^n_{fluid} \tag{24}$$

where $\boldsymbol{RHS}_{fluid}$ includes the right hand side parts of the momentum equations, i.e. the convective, the viscous and the pressure gradient terms.

Finally, the force applied to the body by the flow is calculated through using:

$$\boldsymbol{F} = \sum H\left(\boldsymbol{RHS}_{fluid} - \boldsymbol{f}\right) \tag{25}$$

4.6. Analysis of the numerical diffusion on a 1D flow for both approaches

One of the main difference between weakly-compressible and incompressible schemes apart from the specific role of the pressure, is the upwinding procedure of the convective term discretization. The numerical diffusion introduced by the upwinding is essential to maintain the stability of the scheme. To highlight the difference between the weakly-compressible and incompressible approaches, a brief analysis is performed on a 1D incompressible flow through a planar pipe. The duct section $S(x)$ is not constant and can vary with x as depicted on Fig. 4 for instance.

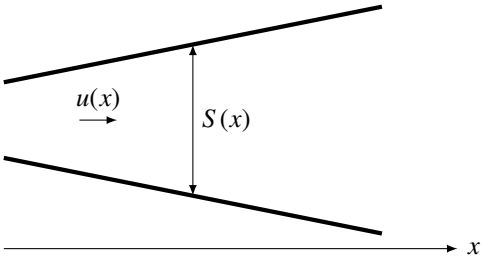


Fig. 4: Quasi 1D flow

The one-dimensional isentropic equation depicting the inviscid flow can be written as follows:

$$\frac{\partial w}{\partial t} + \frac{\partial F(w)}{\partial x} = f(x) \text{ with } w = (\rho S, \rho u S),\ F(w) = (\rho u S, (\rho u^2 + p)S) \text{ and } f = (0, p\frac{dS}{dx}) \tag{26}$$

These equations differ from the standard one dimension Euler equations through the inclusion of the section S. However, the Jacobian matrix remains the same as the original 1D equations.

$$A = \frac{\partial F(w)}{\partial w} = \begin{pmatrix} 0 & 1 \\ c^2 - u^2 & 2u \end{pmatrix} \tag{27}$$

The numerical flux at a given interface can be given as (see Ghidaglia et al. [22, 44]):

$$\phi = \frac{F_L + F_R}{2} - sgn(A)\frac{F_R - F_L}{2} \tag{28}$$

where the subscripts L and R denotes the left and right cells near the interface respectively. In this case, none reconstruction of the fields at the interface for flux computation is taken into account. In practice such procedures without reconstructions are not used since the WENO5 scheme is applied. However, we need this strong assumption to compare analytically the upwinding effect on the numerical diffusion for both solvers. $sgn(A)$ refers to the sign of the Jacobian matrix, i.e. the matrix with the same eigenvectors and the signed eigenvalues. This matrix is actually related to the numerical diffusion introduced to stabilise the scheme.

$$sgn(A) = \frac{sgn(u)}{2}\begin{pmatrix} -Ma & \frac{Ma}{u} \\ u(\frac{1}{Ma} - Ma) & Ma \end{pmatrix} \tag{29}$$

The amount of diffusion ϵ_{wc} introduced in the momentum equation by this upwinding is approximately

$$\epsilon_{wc} = \frac{sgn(u)}{2} u \frac{1}{Ma}\Delta(\rho u) + O(\Delta() \cdot Ma) \tag{30}$$

where Δ is the difference between the left and the right states ($\Delta(\cdot) = (\cdot)_R - (\cdot)_L$).

With a constant density assumption, Eq. (26) can be expressed as:

$$\begin{cases} \dfrac{\partial u}{\partial x} = -\dfrac{u}{S}\dfrac{dS}{dx} \\ \dfrac{\partial \rho u}{\partial t} + \dfrac{\partial(\frac{1}{2}\rho u^2 + p)}{\partial x} = 0 \end{cases} \tag{31}$$

Thus, the numerical flux from the discretization of the convective term is

$$\phi = \frac{(\frac{1}{2}\rho u^2)_L + (\frac{1}{2}\rho u^2)_R}{2} - sgn(u)\frac{(\frac{1}{2}\rho u^2)_R - (\frac{1}{2}\rho u^2)_L}{2} \tag{32}$$

by analogy with the weakly-compressible scheme. The diffusive term is therefore

$$\epsilon_{inc} = \frac{sgn(u)}{2}\Delta(\frac{1}{2}\rho u^2) \approx \frac{sgn(u)}{2} u\Delta(\rho u) \tag{33}$$

So that the ratio of numerical diffusion is

$$\frac{\epsilon_{wc}}{\epsilon_{inc}} \approx \frac{1}{Ma} \approx 10 \text{ under the weakly-compressible assumption with } Ma = 0.1 \tag{34}$$

Strong assumptions have been made here to compare the respective numerical diffusion ($O(Ma)$ terms are neglected, effects of the projection scheme are not taken into account, etc). Nevertheless, this approximation provides an order of magnitude of the amount of diffusion introduced by the upwinding procedure. Perhaps this difference is not relevant for laminar flow due to the importance of physical diffusion. However, it might be more problematic for high Reynolds number flow.

5. Numerical validations

In this section, four test cases illustrate the differences between both formulations in terms of accuracy and efficiency. The computations are performed under the same conditions as far as possible (same spatial resolution and numerical parameters). Computations were run on compute nodes filled with two Intel Xeon E5-2680 v3 Twelve-Core Haswell 2.5GHz processors .

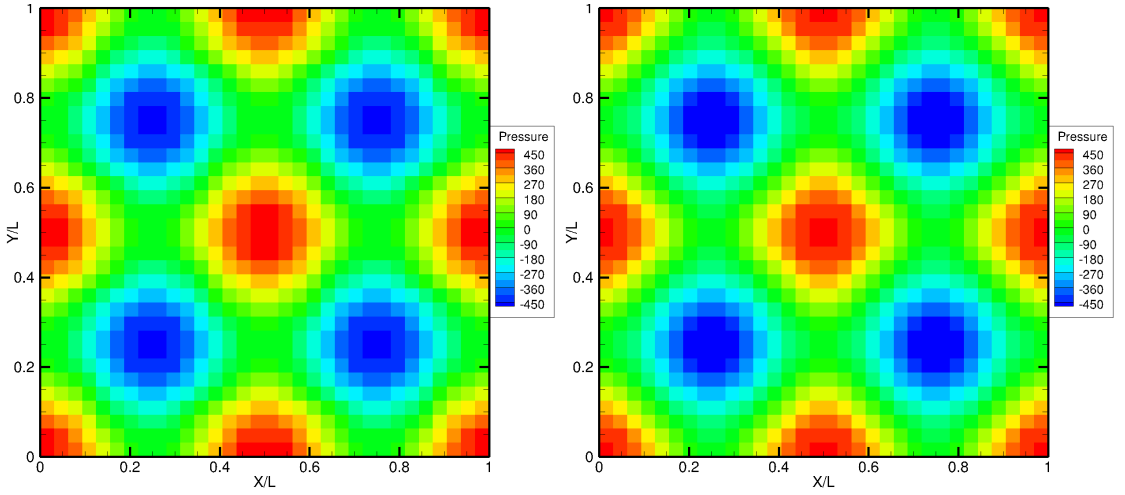
5.1. Numerical diffusion comparison: the 2D Taylor-Green vortex

The Taylor-Green vortex is an incompressible flow of decaying vortices, extensively used to evaluate the numerical diffusion of a scheme. The exact solution of the flow is

$$\begin{cases} u(x,y,t) = U_0 e^{-\frac{8\pi^2\nu t}{L^2}} sin(2\pi\dfrac{x}{L})cos(2\pi\dfrac{y}{L}) \\ v(x,y,t) = -U_0 e^{-\frac{8\pi^2\nu t}{L^2}} cos(2\pi\dfrac{x}{L})sin(2\pi\dfrac{y}{L}) \\ p(x,y,t) = \dfrac{\rho U_0^2}{4} e^{-\frac{16\pi^2\nu t}{L^2}} [cos(4\pi\dfrac{x}{L}) + cos(4\pi\dfrac{y}{L})] \end{cases} \tag{35}$$

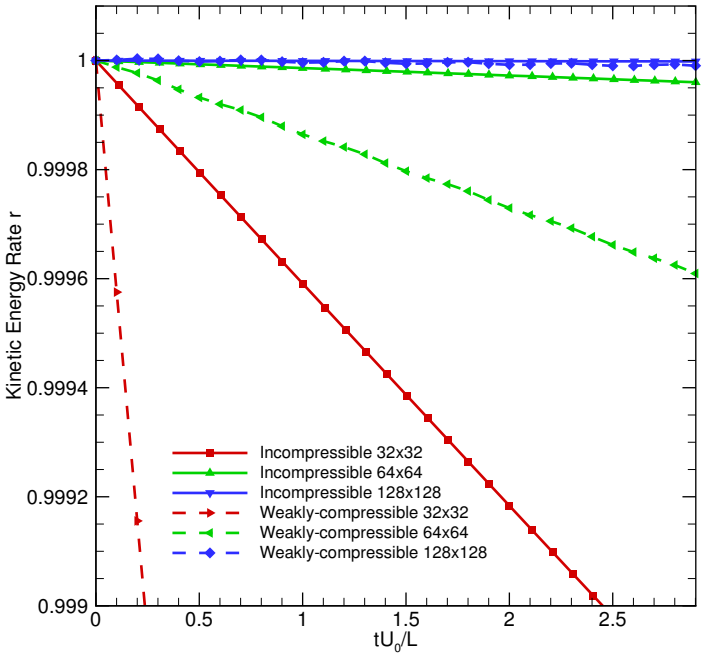
with ν is the kinematic viscosity, $U_0 = 1m/s$ is the reference flow velocity, $\rho = 1000kg/m^3$ is the nominal flow density, and $L = 1m$ is the size of the periodic square domain. Flow is unsteady due to the viscosity. We consider the

inviscid case ($\nu = 0$), i.e. the artificial viscosity introduced by the numerical schemes is only responsible for decaying the vortex amplitude. For the weakly-compressible computation, the nominal sound speed is set to $c_0/U_0 = 10$. The pressure fields at $\frac{tU_0}{L} = 3$ (with 32 cells per direction) are plotted in Fig. 5a and 5b. While both pressure fields look similar, the kinetic energy decay history (Fig. 5c) indicates that the weakly-compressible approach is much more diffusive than the incompressible one. Indeed, it is possible to estimate a numerical viscosity in analogy with a viscous



(a) Weakly-compressible solver

(b) Truly incompressible solver



(c) Kinetic energy decay

Fig. 5: 2D Taylor-Green vortex. Comparison of the pressure fields at time $\frac{tU_0}{L} = 3s$ and kinetic energy decay

flow based on the analytical formula of the decay of the kinetic energy rate $r(t)$:

$$r(t) = \frac{k(t)}{k(t=0)} = \frac{\frac{1}{2}\iint_D \rho(x,y,t)(u^2(x,y,t)+v^2(x,y,t))dxdy}{\frac{1}{2}\iint_D \rho(x,y,0)(u^2(x,y,0)+v^2(x,y,0))dxdy} = e^{-\frac{16\pi^2\nu t}{L^2}} \tag{36}$$

For low diffusion, the decay is a right line, $r(t) \approx 1 - \frac{16\pi^2\nu t}{L^2}$ as seen in Fig. 5c. A linear regression allows for calculation of a numerical viscosity. The estimated numerical diffusions are summarised in Tab. 1 and confirm the theoretical approximation (34).

		Solver	
		Weakly-compressible	Incompressible
Mesh	32x32	2.7×10^{-5}	2.6×10^{-6}
	64x64	8.6×10^{-7}	8.7×10^{-8}
	128x128	2.5×10^{-8}	3.4×10^{-9}

Table 1: 2D Taylor-Green vortex. Numerical viscosity in $m^2.s^{-1}$

The CPU times are also reported in Table 2. Values are expressed in seconds and the computations were run on a single core. It may be noted that the incompressible solver runs around 6 times faster than the weakly-compressible

		Solver	
		Weakly-compressible	Incompressible
Mesh	32x32	1.97 (14.18×10^{-6})	13.55 (8.79×10^{-6})
	64x64	14.56 (13.02×10^{-6})	100.86 (8.16×10^{-6})
	128x128	124.91 (13.91×10^{-6})	796.78 (8.06×10^{-6})

Table 2: 2D Taylor-Green vortex. CPU time and CPU time per time step per cell (in bracket) in s

one while the CPU time per time step per cell is higher. These results are extensively discussed and explained further, for other test cases.

5.2. *Vortex convection in a free-stream*

This second test case also aims to evaluate numerical diffusion. It refers to a benchmark proposed by the CERFACS institute [45] which consists of a single vortex within a free-stream. The initial conditions are defined as

$$\begin{cases} u(x,y) = U_0 + \dfrac{\partial \Phi}{\partial y} \\ v(x,y) = -\dfrac{\partial \Phi}{\partial x} \\ p(x,y) = p_0 - \dfrac{\rho_0 \Gamma^2}{2R_c^2} e^{-\frac{(x-x_c)^2+(y-y_c)^2}{R_c^2}} \end{cases} \tag{37}$$

with $\Phi(x,y) = \begin{cases} \Gamma e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2R_c^2}} & \text{if } \sqrt{(x-x_c)^2+(y-y_c)^2} < 4R_c. \\ 0 \text{ otherwise.} \end{cases}$

$U_0 = 35m/s$ is the free-stream velocity, $L = 0.3112m$ the size of the periodic bounding box, $R_c = \frac{L}{20}$ the radius of

the vortex, $\rho_0 = 1.7170 kg/m^3$ the density and $\Gamma = 0.0359157 m^2.s^{-1}$ is the strength of the vortex. In this case where an inviscid fluid is involved, the initial vortex is supposed to be convected without deformation through the periodic domain. Fig. 6 shows the results obtained on a 80×80 grid after 10 domain crossings. The vortex is better maintained by the incompressible solver, outlining a lower diffusion compared to the weakly-compressible one.

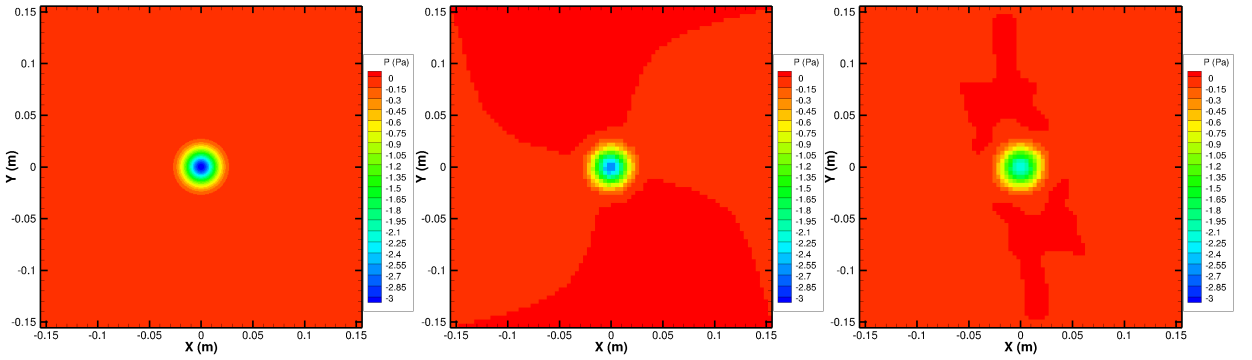


Fig. 6: Vortex convection in a free-stream. Pressure field at $t = 0$ (left) and $t = 10\frac{L}{U_0}$ for a 80×80 grid resolution for the incompressible (middle) and weakly-compressible solver (right).

A spatial convergence study involving various spatial resolutions from 32 to 512 cells per direction is proposed here. The L_2 norm error with the respect to the analytical solution is defined as

$$\begin{cases} \epsilon_{\|\boldsymbol{u}\|_2} = \sqrt{\dfrac{\int_D |\boldsymbol{u} - \boldsymbol{u}_{ref}|^2}{\int_D |\boldsymbol{u}_{ref}|^2}} \\ \epsilon_{\|p\|_2} = \sqrt{\dfrac{\int_D |p - p_{ref}|^2}{\int_D |p_{ref}|^2}} \end{cases} \tag{38}$$

The calculation of the errors is limited to the area of radius $4R_c$. For the incompressible scheme, the analysis is performed on the post-treated pressure from Eq. (17). The results are plotted in Fig. 7. For coarse meshes, the vortex is not sufficiently discretized to define the convergence rate. From 80 cells per direction, the convergence rate of the velocity is greater than 4 for both solvers. The pressure is also at least fourth-order accurate for the weakly-compressible scheme. Surprisingly, a convergence rate close to 4 is approximately obtained while the Pressure Poisson Equation is discretized with a second-order accurate scheme for the incompressible scheme. Saturation is achieved on the finest grids. Note that changing the stopping criterion on the residual of the linear system does not prevent the observed saturation. The convergence testing uses the L_2 norm of the preconditioned residual $P^{-1}(b - Ax)$. By default, the stopping criterion is fixed to a residual value of 10^{-8}. For most of the tested applications, this value is enough to ensure the convergence. It can also be reported that errors for the pressure and velocity for meshes around 350 cells per direction are almost the same for the incompressible and the weakly-compressible schemes. However, the incompressible approach runs 5 times faster than the weakly-compressible approach for this spatial resolution (21.7 hours for the weakly-compressible computation vs 4.05 hours for the incompressible one). The weakly-compressible

solver is the most accurate from 350 cells per direction. On very fine mesh, it should deliver an accurate solution faster than the incompressible solver. This test case illustrates the limitation of the previous reasoning about CPU gain and numerical diffusion.

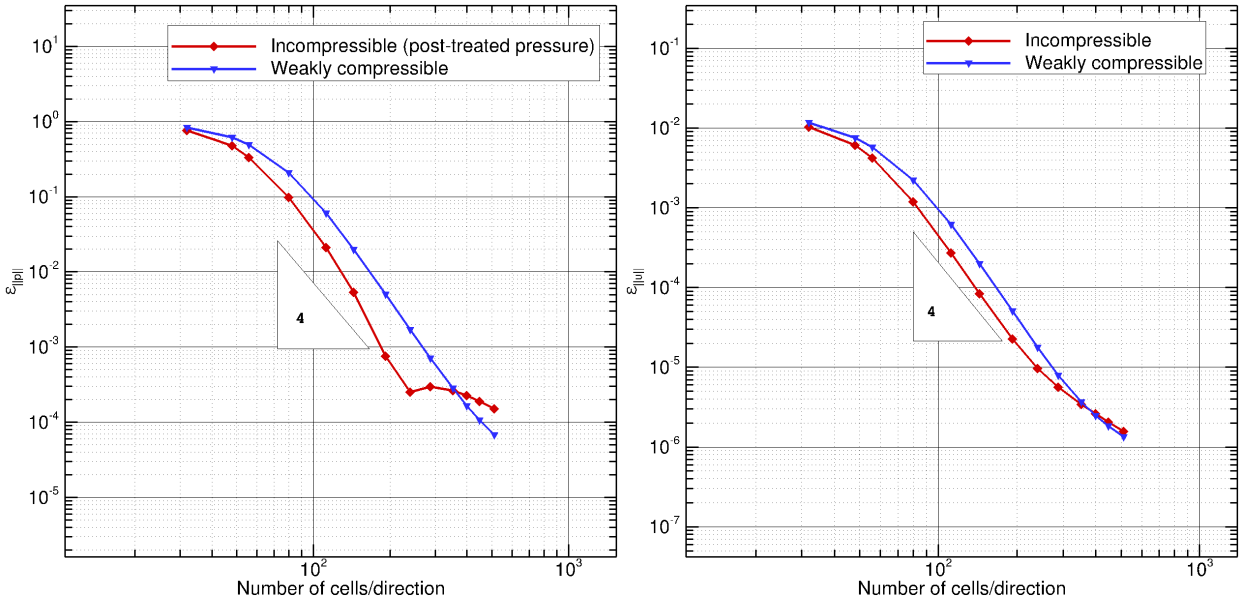


Fig. 7: Vortex convection in a free-stream. Convergence analysis on pressure (left) and velocity (right).

5.3. Taylor-Couette flow

The third test case is the Taylor-Couette flow, involving a 2D incompressible flow between two rotating cylinders. The geometry is described in Fig. 8 and the parameters used in this simulation are:

$$\begin{cases} R_1 = 0.5m \\ \Omega_1 = 1rad/s \\ R_2 = 1m \\ \Omega_2 = 0.25rad/s \end{cases} \tag{39}$$

In this configuration the flow remains perfectly bi-dimensional. The kinematic viscosity is set to $\nu = 1.10^{-3}m^2/s$ and the density to $\rho = 1kg/m^3$. The Reynolds number is defined as $Re = \frac{\Omega_1 R_1^2}{\nu} = 250$. The steady state flow is axisymetric and an analytical solution is available for the orthoradial velocity component and the pressure distribution:

$$\begin{cases} u_\theta(r) = \dfrac{A}{2}r + \dfrac{B}{r} \\ p(r) = p_0 + \rho\left(\dfrac{A^2}{8}r^2 + AB\ln(r) - \dfrac{B^2}{2r^2}\right) \end{cases} \tag{40}$$

where $A = 2\frac{\Omega_2 R_2^2 - \Omega_1 R_1^2}{R_2^2 - R_1^2}$ and $B = -(\Omega_2 - \Omega_1)\frac{R_1^2 R_2^2}{R_2^2 - R_1^2}$. The pressure is defined up to a constant p_0. The computational domain is a bi-dimensional periodic square of length $L = 3m$. For this test case, particular attention is paid to the

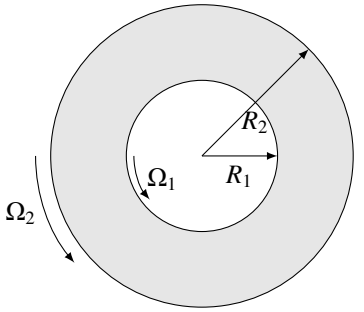


Fig. 8: Taylor-Couette flow

impact of the immersed boundary method onto the overall accuracy. Initially, simulations are run with a uniform Cartesian grid resolution $N_x \times N_y = 64 \times 64$ (Fig. 9). To compare the pressure distributions, constraint (20) is

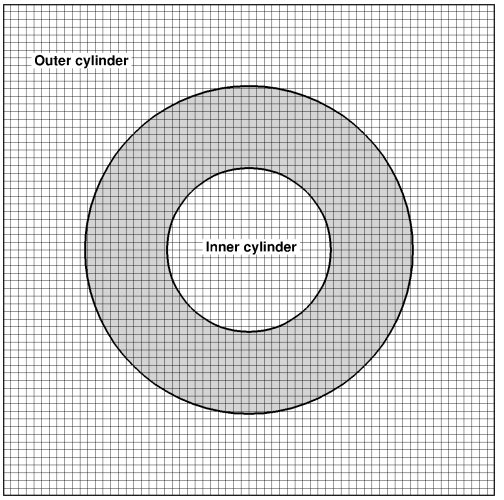


Fig. 9: 2D Taylor-Couette flow. Computational domain for $N_x \times N_y = 64 \times 64$.

applied. The velocity and pressure fields are shown in Fig. 10a,10b and Fig. 10c,10d respectively, which reveals some differences in the vicinity of the inner cylinder.

The velocity and pressure profiles, plotted in Fig. 11a and 11b respectively, indicate that the incompressible solver complies better with the analytical solution. The velocity profiles are similar, including in the solid areas. Note that the pressure inside the inner cylinder is almost constant in the incompressible case. The predicted velocity $\boldsymbol{u}^*$ imposed inside the inner cylinder is a rigid body velocity field. Thus, the pressure equation Eq. (22) reduces to the Laplace equation $\Delta\psi = 0$. Thereby, pressure is a harmonic function inside the inner cylinder. Since pressure is constant at the fluid/solid boundary due to axis symmetry (in accordance with the *maximum principle*), the pressure is expected to be constant everywhere inside the inner cylinder and equal to the pressure value at the boundary. In practice, the pressure is not exactly constant from the boundary but keeps evolving after crossing the boundary from the fluid to the solid side. This is due to the regularized Heaviside step function used to prescribe the velocity inside the body. There is a small area in the vicinity of the boundary where the fluid velocity is solved with the Navier-Stokes

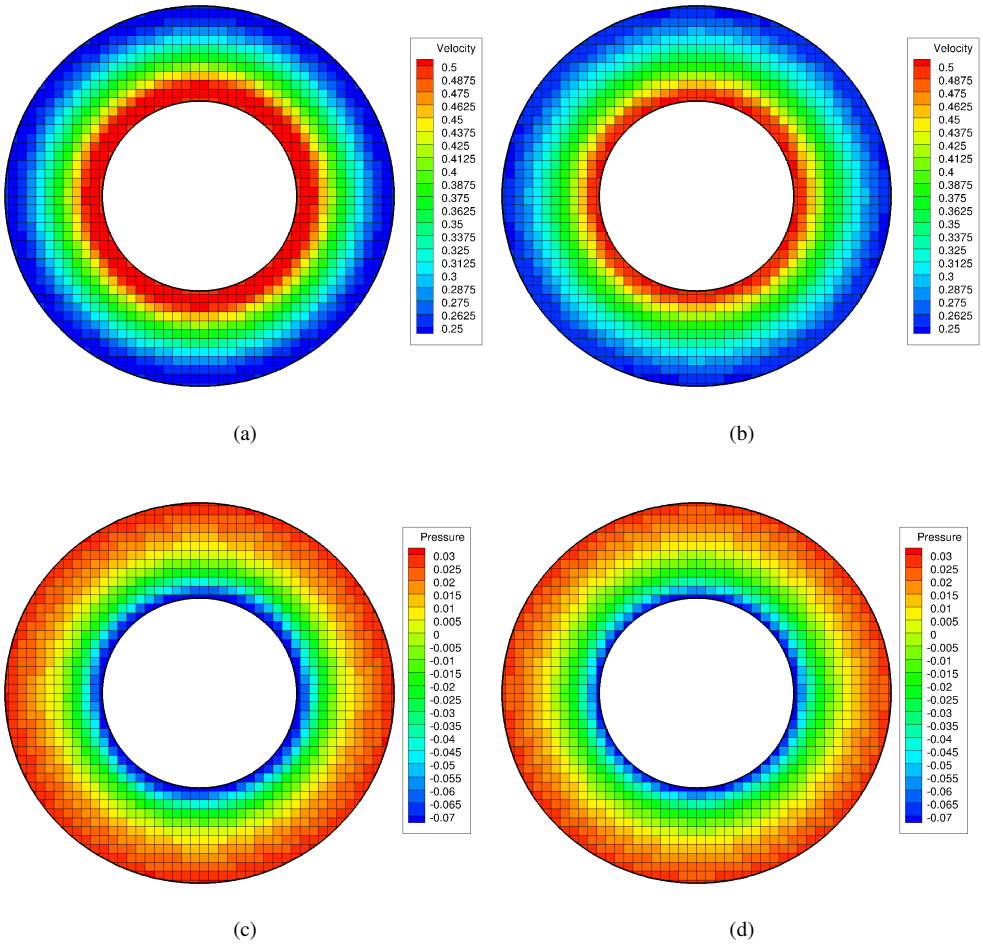


Fig. 10: 2D Taylor-Couette flow. Comparison of velocity and pressure fields at time $t = 230s$ for the weakly-compressible (left) and the truly incompressible (right) solvers.

equations and a solid velocity coexists. The pressure is therefore continuous and differentiable through the boundary. In the weakly-compressible approach, the continuity equation is not restrained with any source term inside the inner cylinder, and follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}_{solid}) = 0 \tag{41}$$

Because the solid motion is rigid without deformation, $\nabla \cdot \boldsymbol{u}_{solid} = 0$ and ρ satisfies the following convection equation:

$$\frac{\partial \rho}{\partial t} + \boldsymbol{u}_{solid} \cdot \nabla \rho = 0 \tag{42}$$

As a consequence, pressure is also expected to be constant inside the cylinder. But in practice, the Riemann solver introduces numerical diffusion that smooths the transition between the pressure fields inside and outside the inner cylinder. Moreover pressure is not expected to vary in time at the centre of the cylinder since $\boldsymbol{u}_{solid} = 0$. It explains the curve of the pressure profile.

Therefore, constant pressure is expected for both weakly-compressible and incompressible approaches. However, a

large variation in pressure has to be captured in the weakly-compressible case. This might be the reason why the pressure profile in the solution to the weakly-compressible approach seems to be further from the analytical solution than the incompressible one.

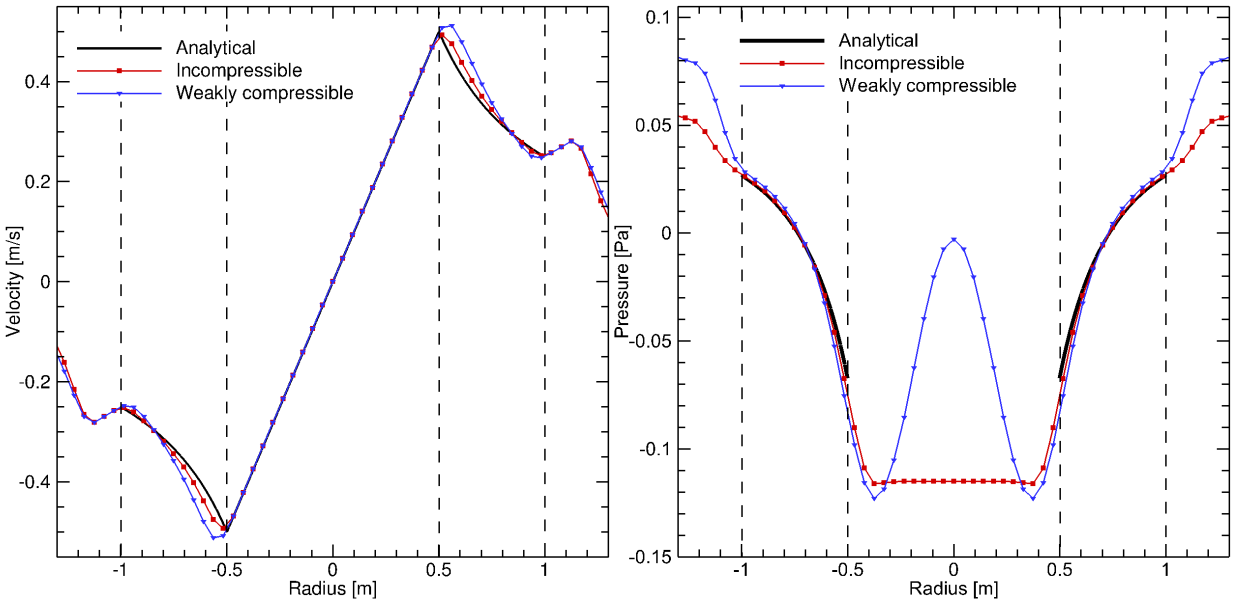


Fig. 11: 2D Taylor-Couette flow. Comparison of the velocity (left) and pressure (right) profiles at time $t = 230s$

To perform a convergence analysis, different spatial resolutions are tested from 32 to 256 cells per direction. The error with respect to the analytical solution is still defined using the L_2 norm described in (38). The convergence rates are almost the same for pressure and velocity. A least-square regression yields approximately a 1.8 convergence rate value for the incompressible solver (Tab. 3) whereas an order of 1.35 is obtained for the weakly-compressible approach (Tab. 4). In addition to a higher convergence rate, the incompressible scheme gives lower error values. Furthermore, this rate is higher than the expected first order. A more limiting measure of the convergence (L_∞ norm for instance), will probably lead to a convergence rate closer to 1. The CPU time versus the error is plotted in Fig. 12a to further investigate the advantages of a truly incompressible scheme. Considering these curves, the CPU time can be written as a function of the error using the following expression:

$$\text{CPU time} = \alpha \epsilon^{\beta} \tag{43}$$

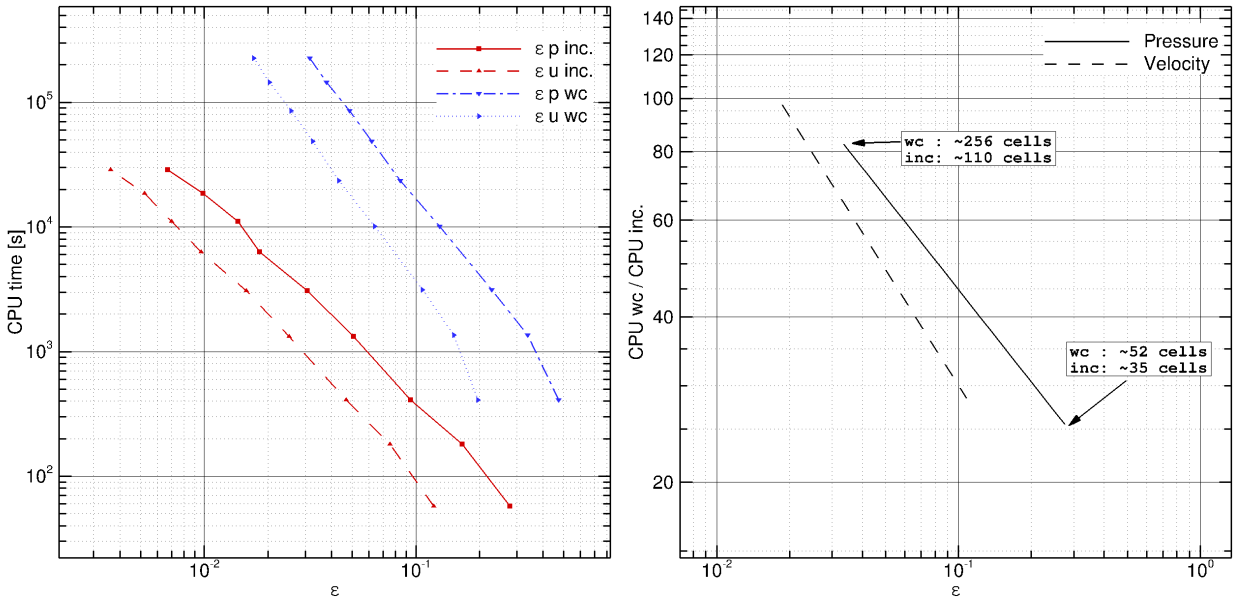
A linear regression of the *CPU time* and ϵ logarithmic allows the coefficients α and β to be deduced for both approaches. It is then possible to determine time saving based on the ratio of the CPU times and depending on the errors using Eq.(43), as shown in Fig. 12b. These curves rule the benefits of the incompressible solver onto the weakly-compressible one at equivalent accuracy. For the example of a target error of $\epsilon = 5\%$, the incompressible solver is approximately 60 times faster than the weakly-compressible one in this test case. This significant gain should be

$N_x \times N_y$	$\epsilon_{\|\boldsymbol{u}\|_2}$	Order	$\epsilon_{\|p\|_2}$	Order
32×32	1.21×10^{-1}	-	2.79×10^{-1}	-
48×48	7.55×10^{-2}	1.17	1.66×10^{-1}	1.28
64×64	4.68×10^{-2}	1.66	9.44×10^{-2}	1.96
96×96	2.52×10^{-2}	1.53	5.08×10^{-2}	1.53
128×128	1.58×10^{-2}	1.62	3.08×10^{-2}	1.75
160×160	9.68×10^{-3}	2.19	1.83×10^{-2}	2.34
192×192	7.03×10^{-3}	1.76	1.44×10^{-2}	1.29

Table 3: 2D Taylor-Couette flow. Spatial convergence analysis for the incompressible solver.

$N_x \times N_y$	$\epsilon_{\|\boldsymbol{u}\|_2}$	Order	$\epsilon_{\|p\|_2}$	Order
32×32	1.97×10^{-1}	-	4.74×10^{-1}	-
48×48	1.51×10^{-1}	0.65	3.39×10^{-1}	0.83
64×64	1.07×10^{-1}	1.19	2.28×10^{-1}	1.37
96×96	6.40×10^{-2}	1.27	1.30×10^{-1}	1.39
128×128	4.32×10^{-2}	1.37	8.43×10^{-2}	1.50
160×160	3.26×10^{-2}	1.26	6.21×10^{-2}	1.37
192×192	2.57×10^{-2}	1.31	4.87×10^{-2}	1.33

Table 4: 2D Taylor-Couette flow. Spatial convergence analysis for the weakly-compressible solver.



(a) (b)

Fig. 12: 2D Taylor-Couette flow. Comparison of the CPU time over the error value (left) and benefit of the incompressible scheme over the weakly-compressible one (right)

carefully considered since the low accuracy of the weakly-compressible solution seems to be linked to the particular pressure field inside the inner cylinder. Much more complex flow might lead to very different conclusions.

5.4. Unsteady flow past a NACA 0010 airfoil at Re=1400

The final test case presented in this work is a laminar flow past a NACA profile at $Re = 1400$ with an incidence angle $\alpha = 30°$. The Reynolds number is defined by :

$$Re = \frac{U_0 c}{\nu} \tag{44}$$

with U_0 is the incident uniform velocity, c is the chord of the airfoil and ν is the kinematic viscosity. The flow is unsteady as a Von Kármán vortex shedding occurs in the wake. The geometry and the configuration of this test case is described in Fig. 13. A locally refined grid is adopted, involving 6 refinement levels.

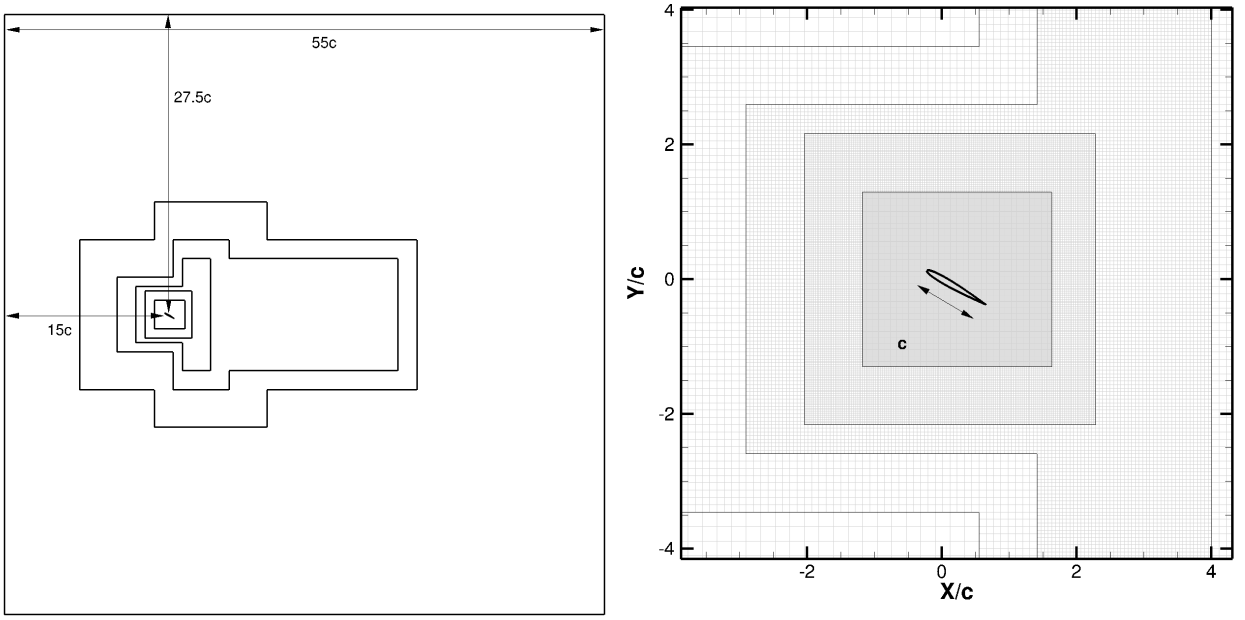
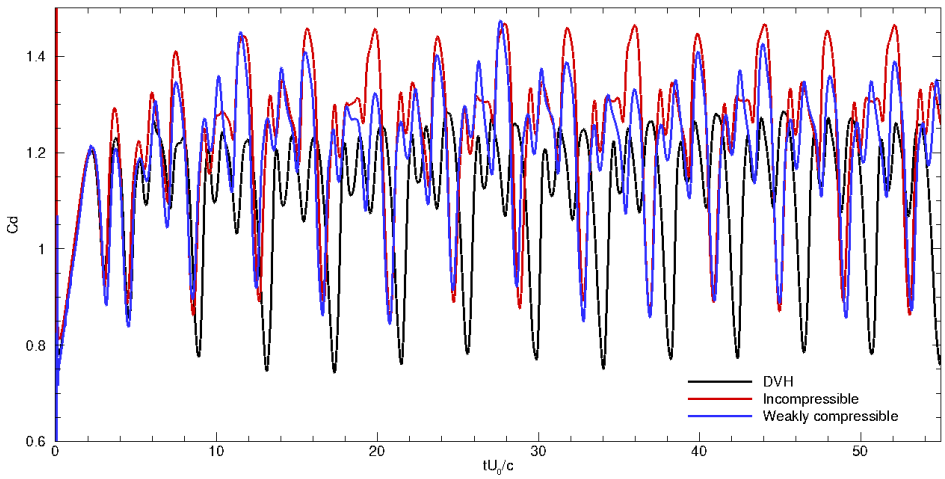


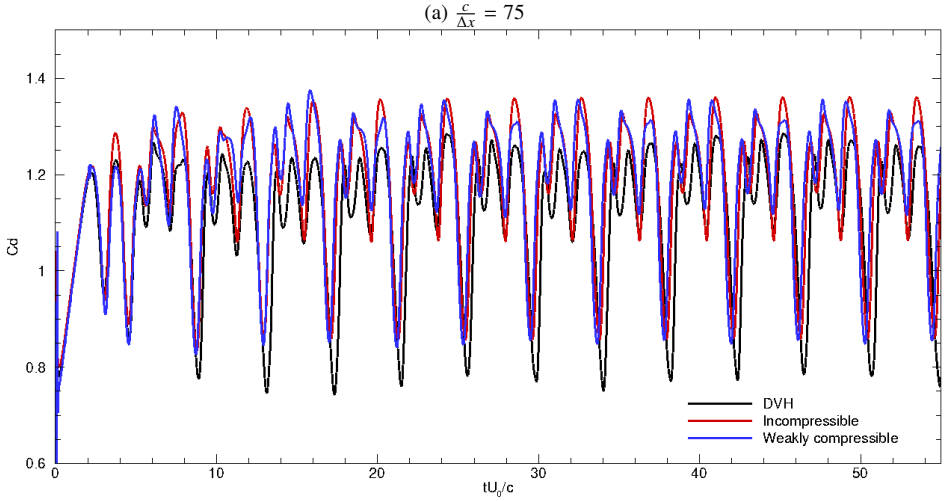
Fig. 13: 2D flow past NACA 0010 airfoil at $Re = 1400$. Computational domain used in the present simulation (left) and locally refined grid in the vicinity of the solid with $\frac{c}{\Delta x} = 75$ (right)

Three spatial resolutions are first studied $\frac{c}{\Delta x} = 75, \frac{c}{\Delta x} = 130$ and $\frac{c}{\Delta x} = 260$. The total number of blocks of the quad-tree is kept the same for all computations. Only the number of cells is adapted to increase the discretization: 8, 14 and 28 cells per direction per block. Once again, computations are run in the same manner and the speed of sound is set to $30U_0$ for the weakly-compressible solver.

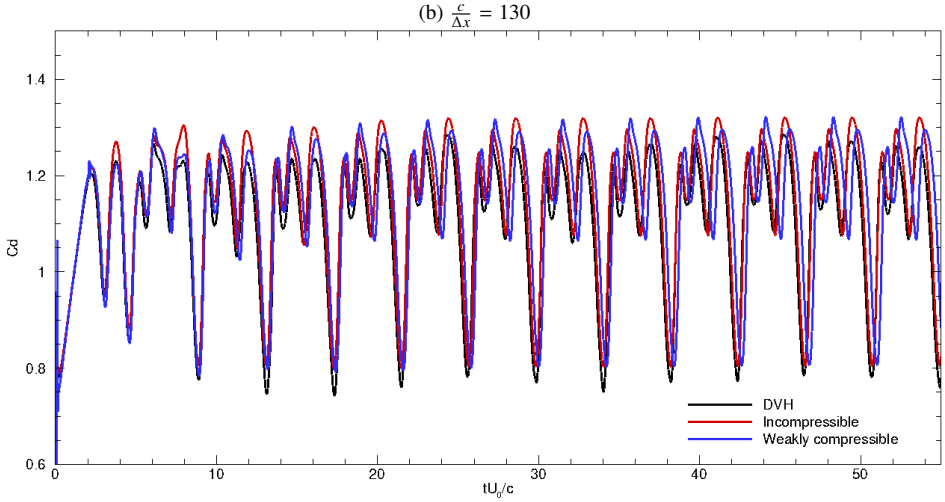
Attention is first payed to the drag coefficient Cd. Fig. 14 presents time history of the drag coefficients. No analytical solution is available for comparing the two solvers. Instead, the results of computations performed with the DVH (Diffused Vortex Hydrodynamic) method developed by Rossi et al. [46] are taken as references. This method is ideally suited for this kind of unsteady laminar flow past complex bodies. The WCCH method has never been successfully compared to DVH method [47]. The spatial resolution of the DVH computation is up to $\frac{c}{\Delta x} = 400$ near the airfoil while it quickly coarsens far away from it. The computations of grids $\frac{c}{\Delta x} = 130$ (Fig. 14b) and $\frac{c}{\Delta x} = 260$ (Fig. 14c) provide some results that converge towards the DVH solution.



(a) $\frac{c}{\Delta x} = 75$



(b) $\frac{c}{\Delta x} = 130$



(c) $\frac{c}{\Delta x} = 260$

Fig. 14: 2D flow past NACA 0010 airfoil at $Re = 1400$. Resulting drag coefficients for different spatial resolutions $\frac{c}{\Delta x} = 75, 130, 260$

The resulting vorticity field is displayed for all computations at $\frac{tU_0}{c} = 100$ (Fig. 15). While the solution of the coarsest mesh seems to be very different from the field given by the DVH, the other two resolutions are better and show that vortex shedding is synchronised with the DVH result. Note that a better solution seems to be better predicted with the incompressible solver than with the weakly-compressible one with the finest mesh.

Fig. 16 shows the pressure field at $\frac{tU_0}{c} = 100$ and confirms the similarity between the various solutions. Here again, the incompressible and weakly-compressible solutions are very similar in terms of accuracy. This is not the case regarding the respective CPU times required.

Table 5 shows the corresponding CPU times as well as the average time step sizes. A factor of almost 10 is observed on the CPU times between incompressible and weakly-compressible schemes. This difference is linked to

Mesh $\frac{c}{\Delta x}$	Total number of cells	Incompressible CPU time (h)	weakly-compressible CPU time (h)	Incompressible Time step (s)	weakly-compressible Time step (s)
75	103552	1.5	15.9	3.20e-03	2.12e-04
130	317128	7.7	87.4	1.85e-03	1.21e-04
260	1268512	65.5	873.5 (estimation)	9.00e-04	6.05e-05

Table 5: 2D flow past NACA 0010 airfoil at $Re = 1400$. Settings and computation time

the time step size which is 15 times larger in the incompressible case. Indeed, the gain of the incompressible compared to the weakly-compressible computation run under the same condition (same mesh, same numerical parameters such as CFL number) can be estimated by:

$$\frac{CPUtime^{WC}}{CPUtime^{inc}} = \frac{t^{WC}_{\Delta t} n^{WC}_{\Delta t}}{t^{inc}_{\Delta t} n^{inc}_{\Delta t}} = \frac{t^{WC}_{\Delta t} \Delta t^{inc}}{t^{inc}_{\Delta t} \Delta t^{WC}} \tag{45}$$

where $t_{\Delta t}$ stands for the CPU time of one time step. Δt is the time step size. $n_{\Delta t}$ is the number of time step. With the same CFL number assumption, the time step size ratio gives:

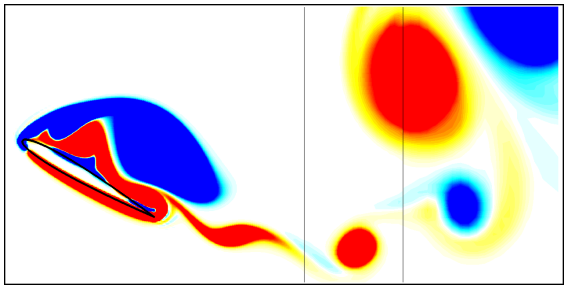
$$\frac{\Delta t^{inc}}{\Delta t^{WC}} = 1 + \frac{c}{|u|_{max}} \tag{46}$$

where c is the speed of sound and u_{max} is the maximum velocity throughout the flow. Considering $c \approx 10 u_{max}$ (i.e. $Ma = 0.1$):

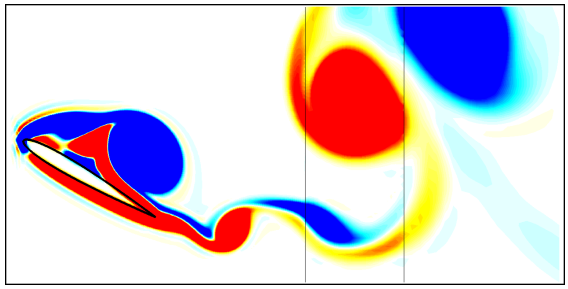
$$\frac{\Delta t^{inc}}{\Delta t^{WC}} \approx 11 \tag{47}$$

This means that the weakly-compressible solver does 11 times more iterations of the time integration scheme (for $Ma = 0.1$).

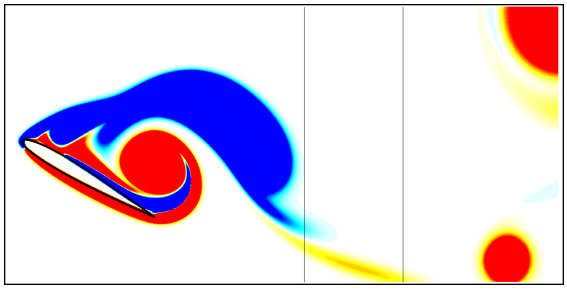
In the present case, the ratio of 15 appears large in comparison to what could be expected according to (47). This is due to the maximum Mach number, which is approximately equal to 0.08. The first few attempts with a sound speed close to $10 u_{max}$ gave less accurate results, so that a higher sound speed and smaller time step was finally chosen. Furthermore, the CPU cost of a single time step is approximately 1.5 higher for the incompressible scheme than for the weakly-compressible one as shown in Fig. 17a, which compares the CPU time per cells and per time step for both approaches with respect to the spatial resolution. Additional spatial resolutions have been added to extend the



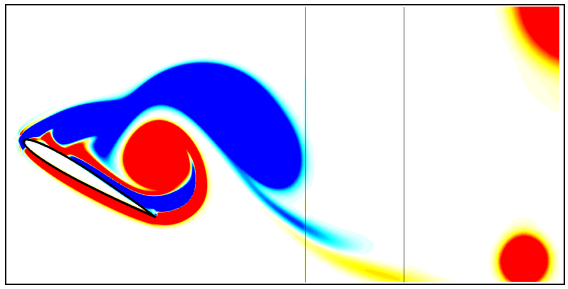
(a) $\frac{c}{\Delta x}$ = 75, incompressible



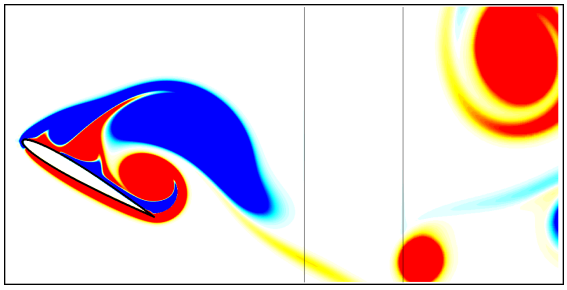
(b) $\frac{c}{\Delta x}$ = 75, weakly-compressible



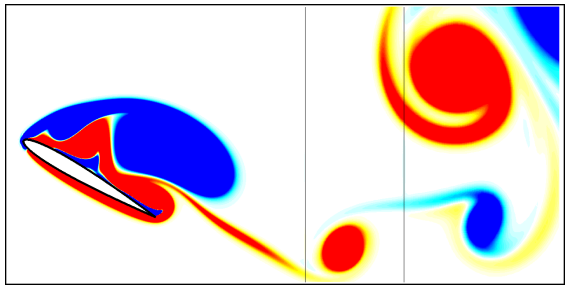
(c) $\frac{c}{\Delta x}$ = 130, incompressible



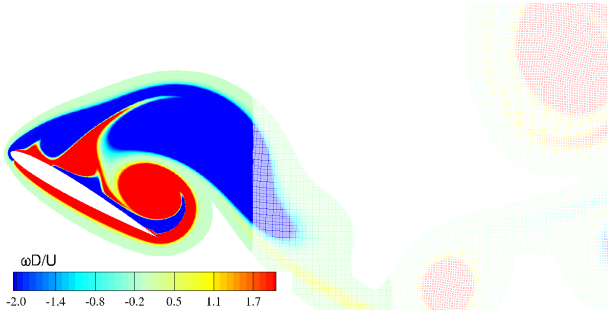
(d) $\frac{c}{\Delta x}$ = 130, weakly-compressible



(e) $\frac{c}{\Delta x}$ = 260, incompressible



(f) $\frac{c}{\Delta x}$ = 260, weakly-compressible



(g) DVH solution with $\frac{c}{\Delta x}$ = 400

Fig. 15: 2D flow past NACA 0010 airfoil at Re = 1400. Vorticity field at $\frac{tU_0}{c}$ = 100. Comparisons between incompressible (left), weakly-compressible (right) and DVH (bottom) solutions.

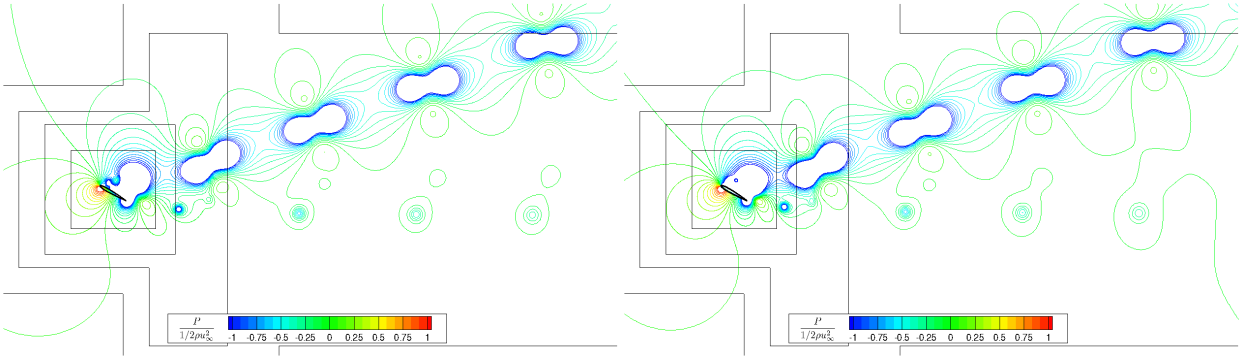


Fig. 16: 2D flow past NACA 0010 airfoil at $Re = 1400$. Pressure field at $\frac{tU_0}{c} = 100$ for $\frac{c}{\Delta x} = 260$. Left : incompressible solver. Right : weakly-compressible solver.

analysis and its sensitivity, from $\frac{c}{\Delta x} = 33$ to $\frac{c}{\Delta x} = 260$ (i.e. 26000 to 1260000 cells). Omitting the first discretization, the CPU time per time step is slightly constant for the incompressible solver. For the weakly-compressible scheme, it is slightly constant up to $\frac{c}{\Delta x} = 185$, before slightly increasing. This result is pertinent and indicates that the computational complexity of the algorithms is global $O(N)$ including the case of the incompressible solver. Finally, according to Fig. 17a the mean ratio of the CPU times for both approaches is:

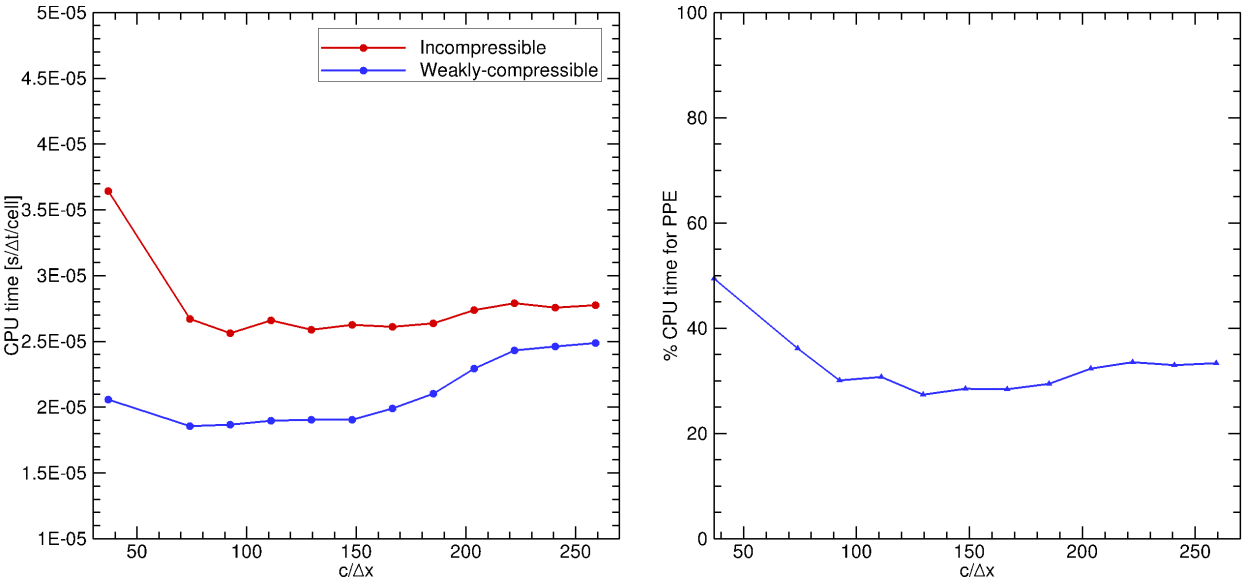
$$\frac{t^{WC}_{\Delta t}}{t^{inc}_{\Delta t}} \approx \frac{1}{1.5} \tag{48}$$

The CPU time of a single time step of the incompressible solver is higher due to the additional time dedicated to solving the Pressure Poisson Equation. The percentage of the CPU time to solve the linear system is plotted in Fig. 17b. This percentage is relatively independent of the number of cells involved and is approximately equal to 30%.

Finally, the incompressible scheme is 10 times faster (15/1.5) at equivalent spatial resolutions, as confirmed in Tab. 5. As performed previously for the Taylor-Couette flow, it is interesting to compare both methods at identical accuracy. In this regard, a convergence analysis is first performed on the drag and lift coefficients for $\frac{c}{\Delta x} = 32$ to 220. The relative error on Cd and Cl (with respect to the DVH solution) is integrated in time from $\frac{tU_0}{c} = 50$ to $\frac{tU_0}{c} = 100$:

$$\epsilon_{Cd} = \sqrt{\int_{t=50}^{t=100} \frac{|Cd - Cd_{DVH}|^2}{Cd^2_{DVH}} dt} \tag{49}$$

We chose this option rather than considering the finest spatial resolution as a reference for each approach in order to compare them to the same solution. Fig. 18a and 18b show the error vs the spatial resolutions for Cd and Cl respectively. Both the weakly-compressible and incompressible solvers converge towards the DVH solution although the convergence curves are not always regular. These results should be carefully considered since the incompressible and weakly-compressible coefficient history are compared to a numerical solution which is not infinitely converged. Note that from $\frac{c}{\Delta x} = 150$ in the incompressible approach and from $\frac{c}{\Delta x} = 180$ in the weakly-compressible case, a rather singular behavior is observed, with the presence of a break in the monotony convergence. That can be explained by looking at the temporal signal plotted in Fig. 14b and 14c. Indeed, Fig. 14c shows that the signal amplitude is better predicted for higher spatial resolutions. However, a larger temporal shifting can also be noticed with respect

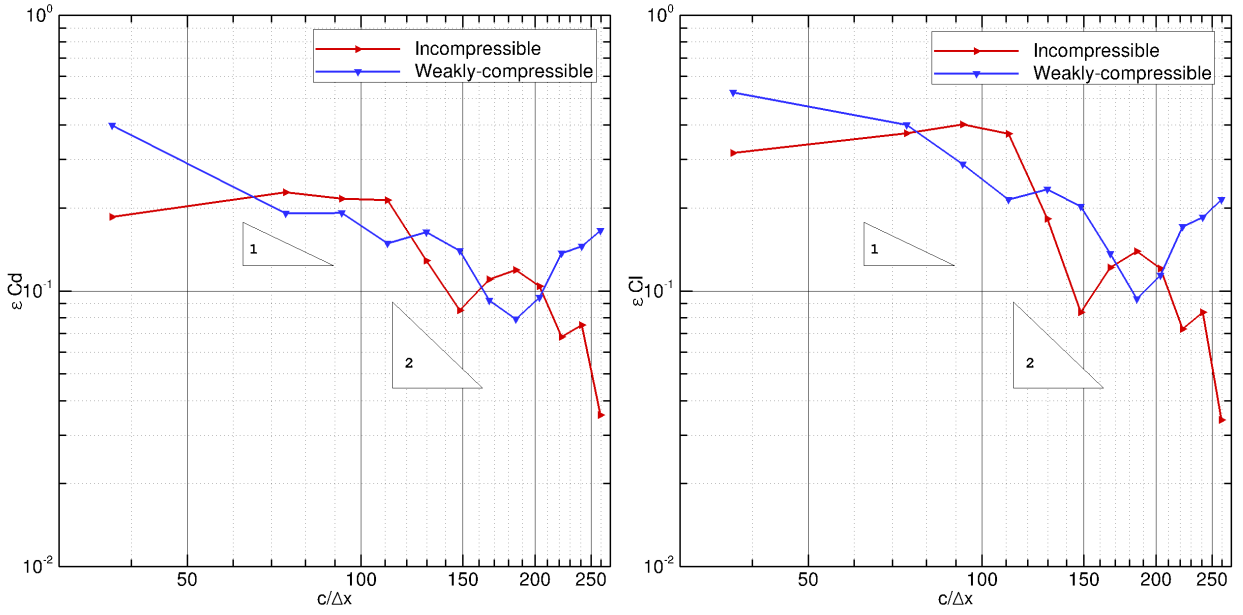


(a) CPU time per time step and per cell.

(b) Percentage of the total CPU time dedicated to solving Pressure Poisson Equation of a single time step.

Fig. 17: 2D flow past NACA 0010 airfoil at Re = 1400. CPU informations

to the medium spatial resolution (see Fig. 14b) explaining this sudden change in the monotony convergence. Further investigations will be conducted to better understand the origin of this frequency shifting on fine grids. Conversely



(a) (b)

Fig. 18: 2D flow past NACA 0010 airfoil at Re = 1400. Convergence analysis on Cd (left) and Cl (right).

to the previous test case, it seems impossible to evaluate the CPU time here as an analytical function of the error using a linear regression. The CPU time savings observed in this case are much lower than the ones obtained for the Taylor-Couette flow, but are still significant; the incompressible scheme always seems to be faster than the weakly-compressible one for a given accuracy with a time saving factor between 5 and 20 for medium and fine meshes.

This test case illustrates the suitability of the truly incompressible approach to solve an unsteady flow past a solid body in comparison with the weakly-compressible solver.

6. Discussion on High-Performance Computing elements

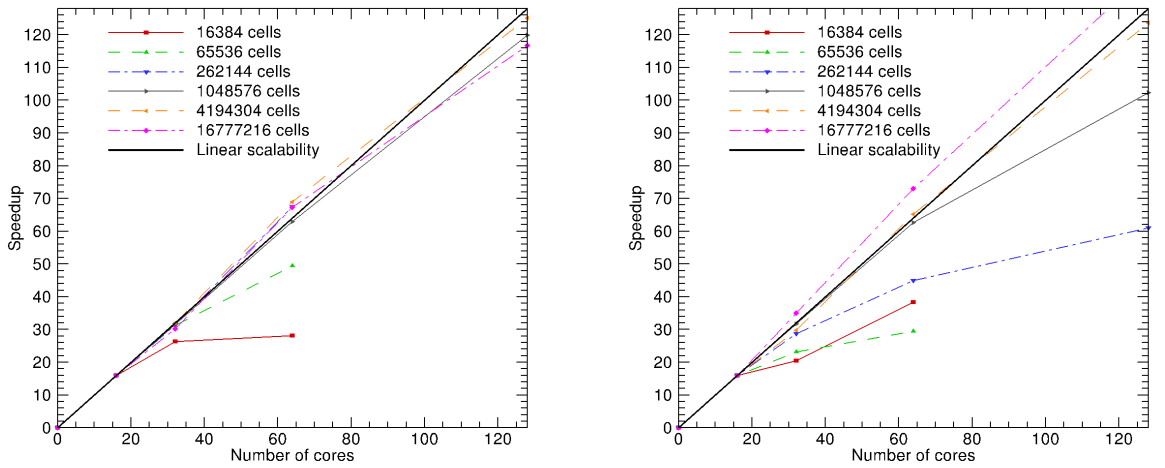
6.1. Parallelization efficiency

The present developments were integrated in the parallel framework of the weakly-compressible solver. The message-passing paradigm MPI is employed to run the solver on distributed memory parallel computers. Parallelization is made efficient with a domain decomposition strategy and an overlapping of local computations by MPI communications. The weakly-compressible code demonstrated strong scalability on static meshes without AMR. The main feature introduced by the incompressible solver is the resolution of the Poisson equation, which can potentially affect the parallel efficiency. Iterative methods used to solve such a linear system are scalable for one iteration step most of the time. However, the number of iterations necessary to sufficiently decrease the residual increases with the size of the system due to a slow convergence of low spatial frequencies. Preconditioning or multigrid methods speed up the resolution and allow convergence in few iterations. For the past decade, active research has been performed to make algebraic multigrid (AMG) methods as competitive as geometric multigrid approaches on massively parallel computers. BoomerAMG, as part of the HYPRE library, is a robust and parallel implementation of AMG [48, 49] and demonstrates excellent scalability on extreme-scale parallel supercomputers. Therefore, no significant deterioration is expected on the solver. A 2D flow past a cylinder at Reynolds number 40 on a uniform grid is selected as the test case to perform the scalability study. Computations were ran up to 10 compute nodes with two Intel Six-Core Westmere EP 2.8GHz processors connected with Infiniband switch interconnects. Fig. 19 show the scalability curves. The incompressible solver scalability (Fig. 19a) is not as perfect as the original compressible solver (Fig. 19b), especially when the number of cells per node is low. A super-linear speedup is also obtained for the computation with 16 millions of cells. This behaviour can be explained by the presence of favorable cache effects, due to a decrease of the buffer sizes as the number of cores involved increases.

Such observations have been already made by Almgren et al. in [50]. They compared two codes based on the AMR library BoxLib [51]: MAESTRO, a hydrodynamics code based on a low Mach number approach with a linear system to solve and CASTRO, a fully explicit compressible code. CASTRO has a better scaling performance than MAESTRO, which is a bit far from an ideal linear speedup.

6.2. Does the incompressible solver is always faster than the weakly-compressible one?

As shown in Section 5, the incompressible scheme provides certain benefits over the weakly-compressible approach at equivalent accuracy. However, the previous conclusions are related to the current implementation of the



(a) (b)

Fig. 19: Speedup analysis. Comparison of the pressure fields at time $t = 3s$ for the weakly-compressible solver (left) and the truly incompressible solver (middle). Kinetic energy rate dissipation (right)

algorithms using our in-house solver. It raises the following question: are the previous conclusions unchanged if the weakly-compressible solver is better optimized?

In practice accelerating the weakly-compressible code of a given factor is not enough to be assured that the CPU gain of the incompressible scheme over the weakly-compressible one would be divided by this factor. Indeed, the prediction step of the Chorin's projection is very similar to the explicit step of the WCCH method. In our solver which contains the incompressible and the weakly-compressible schemes, these related algorithms are shared by both of them. One can consider a current gain of the incompressible solver over the weakly-compressible one for a given accuracy in the range $[5, 20]$. These values are the ones encountered in the simulation of the flow past the NACA airfoil. The effective speedup factor of the algorithms of the weakly-compressible approach would be higher in reality, in the range $[9, 70]$ following an estimation presented in Appendix A.

Furthermore, these factors are linked to the percentage of the CPU time dedicated to the Poisson equation for the incompressible solver. The speedup factor to reach is lower if the percentage of the CPU time spent to solve the Pressure Poisson Equation is higher as demonstrated in Appendix A. Perhaps the conclusions would be definitely different if the prediction step used a low-order numerical schemes like the MUSCL scheme for the reconstructions step in the convective term discretization, instead of the WENO5 scheme.

Hence, the algorithms of the explicit part of the solver have to be very fast and efficient to make the WCCH method more competitive than the incompressible approach. For now, reaching the speedup factors mentioned in Appendix A in our solver appears to be a challenging task.

7. Conclusion

In this paper, incompressible and weakly-compressible schemes for hydrodynamic flows have been compared within the Finite Volume framework. Computations have been performed on four different test cases to highlight strengths and weaknesses of each approach.

From a computational point of view, the weakly-compressible solver requires less computational efforts. Its implementation is algorithmically straightforward with a low memory footprint and a highly scalable behavior. On the other hand, the truly incompressible scheme accepts larger time steps and appears to be more accurate. In our case, it has been observed that the incompressible formulation can be up to 20 times faster in some test cases for an equivalent targeted accuracy for both approaches. It should be noted that these results apply to a specific context of high-order schemes and Cartesian grid-based methods.

The main drawbacks of the incompressible formulation are a lower scalability while decreasing the number of cells per core and a more complex algorithmics.

Looking ahead, additional investigations on the comparisons of weakly-compressible and incompressible approaches considering high Reynolds number or multiphase flow could be performed to confirm or refute the aforementioned conclusions. Future investigations might also consider comparisons of the two approaches regarding dynamic adaptive mesh refinement.

References

[1] P. Bigay, G. Oger, P.-M. Guilcher, D. Le Touzé, A weakly-compressible cartesian grid approach for hydrodynamic flows, Computer Physics Communications 220 (2017) 31–43.
[2] E. Turkel, Preconditioned methods for solving the incompressible and low speed compressible equations, Journal of computational physics 72 (1987) 277–298.
[3] H. Guillard, C. Viozat, On the behaviour of upwind schemes in the low mach number limit, Computers & fluids 28 (1999) 63–86.
[4] J. J. Monaghan, Simulating free surface flows with SPH, Journal of computational physics 110 (1994) 399–406.
[5] G. Oger, D. Le Touzé, D. Guibert, M. De Leffe, J. Biddiscombe, J. Soumagne, J.-G. Piccinali, On distributed memory mpi-based parallelization of sph codes in massive hpc context, Computer Physics Communications 200 (2016) 1–14.
[6] A. J. Chorin, Numerical solution of the navier-stokes equations, Mathematics of computation 22 (1968) 745–762.
[7] R. Temam, Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (ii), Archive for Rational Mechanics and Analysis 33 (1969) 377–385.
[8] F. H. Harlow, J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, The physics of fluids 8 (1965) 2182–2189.
[9] L. Caretto, A. Gosman, S. Patankar, D. Spalding, Two calculation procedures for steady, three-dimensional flows with recirculation, in: Proceedings of the third international conference on numerical methods in fluid mechanics, Springer, 1973, pp. 60–68.
[10] S. Patankar, Numerical heat transfer and fluid flow, CRC press, 1980.
[11] R. I. Issa, Solution of the implicitly discretised fluid flow equations by operator-splitting, Journal of computational physics 62 (1986) 40–65.
[12] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries, Journal of Computational Physics 190 (2003) 572–600.
[13] B. E. Griffith, R. D. Hornung, D. M. McQueen, C. S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, Journal of computational physics 223 (2007) 10–49.
[14] E.-S. Lee, C. Moulinec, R. Xu, D. Violeau, D. Laurence, P. Stansby, Comparisons of weakly compressible and truly incompressible algorithms for the sph mesh free particle method, Journal of computational physics 227 (2008) 8417–8436.
[15] J. P. Hughes, D. I. Graham, Comparison of incompressible and weakly-compressible sph models for free-surface water flows, Journal of Hydraulic Research 48 (2010) 105–117.
[16] M. S. Shadloo, A. Zainali, M. Yildiz, A. Suleman, A robust weakly compressible sph method and its comparison with an incompressible sph, International Journal for Numerical Methods in Engineering 89 (2012) 939–956.
[17] A. J. Chorin, A numerical method for solving incompressible viscous flow problems, Journal of computational physics 2 (1967) 12–26.
[18] C. MERKLE, Time-accurate unsteady incompressible flow algorithms based on artificial compressibility, in: 8th Computational Fluid Dynamics Conference, 1987, p. 1137.
[19] P. Tamamidis, G. Zhang, D. N. Assanis, Comparison of pressure-based and artificial compressibility methods for solving 3d steady incompressible viscous flows, Journal of Computational Physics 124 (1996) 1–13.

[20] R. L. Bevan, E. Boileau, R. van Loon, R. Lewis, P. Nithiarasu, A comparative study of fractional step method in its quasi-implicit, semi-implicit and fully-explicit forms for incompressible flows, International Journal of Numerical Methods for Heat & Fluid Flow 26 (2016) 595–623.
[21] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted eno schemes, Journal of computational physics 126 (1996) 202–228.
[22] J.-M. Ghidaglia, A. Kumbaro, G. Le Coq, Une méthode volumes finis à flux caractéristiques pour la résolution numérique des systèmes hyperboliques de lois de conservation, Comptes rendus de l'Académie des sciences. Série 1, Mathématique 322 (1996) 981–988.
[23] J. Kim, P. Moin, Application of a fractional-step method to incompressible navier-stokes equations, Journal of computational physics 59 (1985) 308–323.
[24] H. Le, P. Moin, An improvement of fractional step methods for the incompressible navier-stokes equations, Journal of computational physics 92 (1991) 369–379.
[25] J. M. Pereira, M. Kobayashi, J. C. Pereira, A fourth-order-accurate finite volume compact method for the incompressible navier–stokes solutions, Journal of Computational Physics 167 (2001) 217–243.
[26] N. A. Kampanis, J. A. Ekaterinaris, A staggered grid, high-order accurate method for the incompressible navier–stokes equations, Journal of Computational Physics 215 (2006) 589–613.
[27] D. L. Brown, R. Cortez, M. L. Minion, Accurate projection methods for the incompressible navier–stokes equations, Journal of computational physics 168 (2001) 464–499.
[28] J. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Computer methods in applied mechanics and engineering 195 (2006) 6011–6045.
[29] B. Sanderse, B. Koren, Accuracy analysis of explicit runge–kutta methods applied to the incompressible navier–stokes equations, Journal of Computational Physics 231 (2012) 3041–3063.
[30] L. Vittoz, G. Oger, Z. Li, M. de Leffe, D. Le Touzé, A High-Order Finite Volume Solver on Locally Refined Cartesian Meshes-Benchmark Session, Springer International Publishing, 2017, pp. 73–89.
[31] A. S. Almgren, J. B. Bell, W. G. Szymczak, A numerical method for the incompressible navier-stokes equations based on an approximate projection, SIAM Journal on Scientific Computing 17 (1996) 358–369.
[32] A. S. Almgren, J. B. Bell, W. Y. Crutchfield, Approximate projection methods: Part i. inviscid analysis, SIAM Journal on Scientific Computing 22 (2000) 1139–1159.
[33] S. Popinet, Basilisk, a free software program for the solution of partial differential equations on adaptive cartesian meshes, `http://basilisk.fr`, 2015. Accessed:2017-09-01.
[34] J. B. Bell, P. Colella, H. M. Glaz, A second-order projection method for the incompressible navier-stokes equations, Journal of Computational Physics 85 (1989) 257–283.
[35] M. L. Minion, et al., Semi-implicit spectral deferred correction methods for ordinary differential equations, Communications in Mathematical Sciences 1 (2003) 471–500.
[36] M. L. Minion, Semi-implicit projection methods for incompressible flow based on spectral deferred corrections, Applied numerical mathematics 48 (2004) 369–387.
[37] A. S. Almgren, A. Aspden, J. B. Bell, M. L. Minion, On the use of higher-order projection methods for incompressible turbulent flow, SIAM Journal on Scientific Computing 35 (2013) B25–B42.
[38] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.
[39] R. Falgout, U. Yang, hypre: A library of high performance preconditioners, Computational Science—ICCS 2002 (2002) 632–641.
[40] P. MacNeice, K. M. Olson, C. Mobarry, R. De Fainchtein, C. Packer, Paramesh: A parallel adaptive mesh refinement community toolkit, Computer physics communications 126 (2000) 330–354.
[41] E. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, Journal of Computational Physics 161 (2000) 35–60.
[42] T. Kajishima, S. Takiguchi, Interaction between particle clusters and particle-induced turbulence, International Journal of Heat and Fluid Flow 23 (2002) 639–646.
[43] K. Goda, A multistep technique with implicit difference schemes for calculating two-or three-dimensional cavity flows, Journal of Computational Physics 30 (1979) 76–95.
[44] J.-M. Ghidaglia, Flux schemes for solving nonlinear systems of conservation laws, Innovative Methods for Numerical Solution of Partial Differential Equations, Chattot JJ and Hafez M. Eds (2001) 232–242.
[45] CERFACS, The cerfacs co-vo test (convection of a vortex) for dns and les codes, `http://elearning.cerfacs.fr/numerical/benchmarks/vortex2d`, 2011. Accessed:2017-09-01.
[46] E. Rossi, A. Colagrossi, B. Bouscasse, G. Graziani, The diffused vortex hydrodynamics method, Communications in Computational Physics 18 (2015) 351–379.
[47] E. Rossi, A. Colagrossi, G. Oger, D. Le Touzé, Multiple bifurcations of the flow over stalled airfoils when changing the reynolds number, Journal of Fluid Mechanics 846 (2018) 356–391.
[48] U. M. Yang, et al., Boomeramg: a parallel algebraic multigrid solver and preconditioner, Applied Numerical Mathematics 41 (2002) 155–177.
[49] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, J. W. Ruge, Robustness and scalability of algebraic multigrid, SIAM Journal on Scientific Computing 21 (2000) 1886–1908.
[50] A. Almgren, J. Bell, D. Kasen, M. Lijewski, A. Nonaka, P. Nugent, C. Rendleman, R. Thomas, M. Zingale, Maestro, castro, and sedona–petascale codes for astrophysical applications, arXiv preprint arXiv:1008.2801 (2010).
[51] J. Bell, A. Almgren, V. Beckner, M. Day, M. Lijewski, A. Nonaka, W. Zhang, Boxlib user's guide, github. com/BoxLib-Codes/BoxLib (2012).

Appendix A. Impact study of the acceleration or optimization of the algorithms of the explicit part of the solver

In the following, two studies are performed with an hypothetical speedup of the weakly-compressible solver (which also concerns the explicit part of the incompressible solver): one considering the same spatial resolution for both solvers and another one with the same targeted level of accuracy.

Appendix A.1. Effects of an optimization at same spatial resolution

First, the effect of speeding up the weakly-compressible scheme on the CPU benefit of the incompressible formulation is studied (with identical grid). The CPU benefit of the incompressible scheme over the weakly-compressible one is :

$$\frac{t^{wc}_{CPU}}{t^{inc}_{CPU}} = \frac{t^{wc}_{CPU/\Delta t}}{t^{inc}_{CPU/\Delta t}} \frac{\Delta t^{wc}}{\Delta t^{inc}} \tag{A.1}$$

where t_{CPU} is the total CPU time, $t_{CPU/\Delta t}$ is the CPU time for a single time step and Δt is the time step size. Considering the same CFL number for both solvers, Eq. (47) gives:

$$\frac{\Delta t^{wc}}{\Delta t^{inc}} = 1 + \frac{1}{Ma} \tag{A.2}$$

Furthermore, in a simple way, the CPU time of one-time step of the incompressible solver can be decomposed into an explicit part (prediction of intermediate velocity $\boldsymbol{u}^*$) and a time dedicated part to solve the linear system. The first part is very similar in both codes and uses almost the same algorithms.

$$t^{inc}_{CPU/\Delta t} = \alpha t^{wc}_{CPU/\Delta t} + t^{Poisson}_{CPU/\Delta t} \tag{A.3}$$

α is a weight coefficient due to the fact that the velocity prediction within the Chorin's projection and the weakly-compressible scheme are not exactly the same due to the differences in variable arrangement. It also takes into account the Riemann problem solved exclusively in the weakly-compressible scheme. Some numerical experiments have shown that $\alpha \approx 1.0$ but values between $0.8 \leq \alpha \leq 1.2$ are selected in the following. Finally, the CPU time for the Pressure Poisson Equation is only a part p of the total CPU time within a single time step:

$$t^{Poisson}_{CPU/\Delta t} = p t^{inc}_{CPU/\Delta t} \tag{A.4}$$

As seen $p \approx 0.3$ but values can reasonably be chosen in the range $0.3 \leq p \leq 0.5$. Finally:

$$t^{Poisson}_{CPU/\Delta t} = \alpha \frac{p}{1-p} t^{wc}_{CPU/\Delta t} \tag{A.5}$$

Thus, the CPU benefit is

$$\frac{t^{wc}_{CPU}}{t^{inc}_{CPU}} = \frac{t^{wc}_{CPU/\Delta t}}{\alpha t^{wc}_{CPU/\Delta t} + \alpha \frac{p}{1-p} t^{wc}_{CPU/\Delta t}} (1 + \frac{1}{Ma}) \tag{A.6}$$

The speedup factor γ defines the speedup of the weakly-compressible solver by optimizing the algorithms. The new CPU cost of a single time step is now:

$$\tilde{t}^{wc}_{CPU/\Delta t} = \frac{1}{\gamma} t^{wc}_{CPU/\Delta t} \tag{A.7}$$

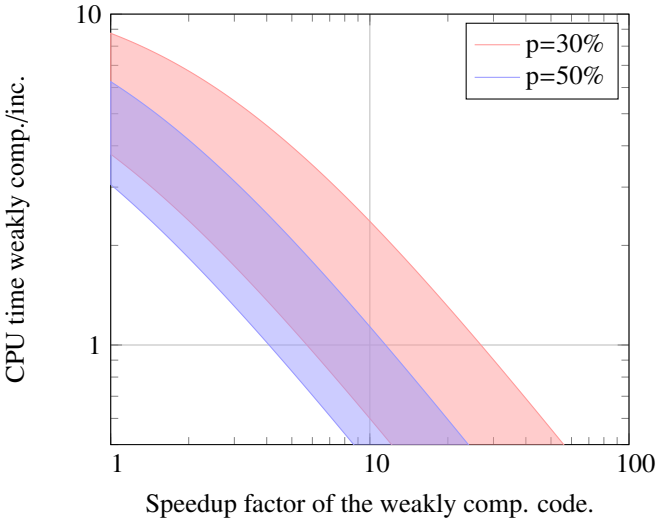


Fig. A.20: CPU benefit of the incompressible/weakly-compressible schemes vs speedup of the weakly-compressible scheme (with same grid).

$\gamma > 1$ if the CPU time is decreased. The new CPU benefits is given by:

$$\frac{t^{wc}_{CPU}}{t^{inc}_{CPU}} = \frac{\tilde{t}^{wc}_{CPU/\Delta t}}{\alpha \tilde{t}^{wc}_{CPU/\Delta t} + \alpha \frac{p}{1-p} t^{wc}_{CPU/\Delta t}} (1 + \frac{1}{Ma}) \tag{A.8}$$

Indeed, the optimization of weakly-compressible algorithms also benefits the incompressible solver. The resolution of the Pressure Poisson Equation with PETSc is not affected. After simplifying:

$$\frac{t^{wc}_{CPU}}{t^{inc}_{CPU}} = \frac{1}{\alpha(1 + \gamma \frac{p}{1-p})} (1 + \frac{1}{Ma}) \tag{A.9}$$

The figure A.20 plots the CPU factor $\frac{t^{wc}_{CPU}}{t^{inc}_{CPU}}$ as a function of the speedup factor γ. The Mach number is set to 0.1. Two values of p are selected $p = 0.3$ and $p = 0.5$. α is varied in the range $[0.8; 1.2]$, which explains the thick shape of the curves. We therefore observe from this plot that the weakly-compressible scheme should be accelerated from a factor between 4 and 30 to generate CPU times computation similar to those of the incompressible one.

Appendix A.2. Effects of optimization at same accuracy

The initial CPU gain before any hypothetical optimization is noted as G:

$$G = \frac{t^{wc}_{CPU/\Delta t}}{t^{inc}_{CPU/\Delta t}} \frac{\Delta t^{wc}}{\Delta t^{inc}} \tag{A.10}$$

or

$$G = \frac{t^{wc}_{CPU/\Delta t}}{\alpha t^{wc}_{CPU/\Delta t} + \alpha \frac{p}{1-p} t^{wc}_{CPU/\Delta t}} \frac{\Delta t^{wc}}{\Delta t^{inc}} \tag{A.11}$$

$$G = \frac{\alpha}{1-p} \frac{\Delta t^{wc}}{\Delta t^{inc}} \tag{A.12}$$

The ratio $\frac{\Delta t^{wc}}{\Delta t^{inc}}$ can be expressed in terms of G:

$$\frac{\Delta t^{wc}}{\Delta t^{inc}} = G \frac{1-p}{\alpha} \tag{A.13}$$

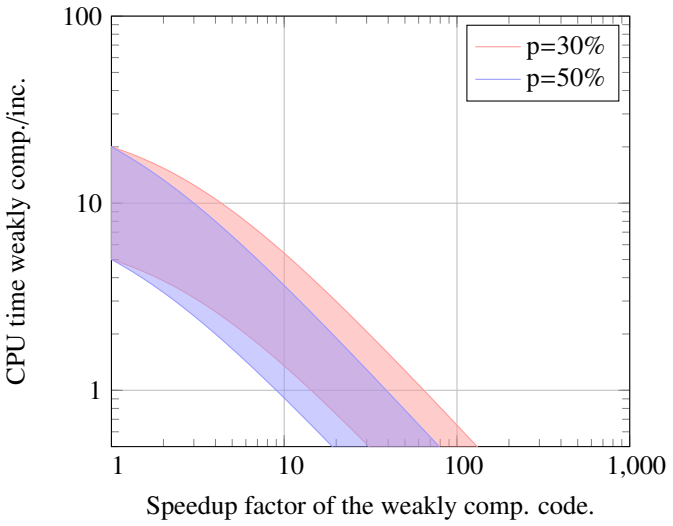


Fig. A.21: CPU benefit of the incompressible/weakly-compressible schemes vs speedup of the weakly-compressible scheme (with same accuracy).

Following the previous study, the new CPU factor after optimizing the weakly-compressible code is:

$$\frac{t^{wc}_{CPU}}{t^{inc}_{CPU}} = \frac{1}{\alpha(1+\gamma\frac{p}{1-p})}\frac{\Delta t^{wc}}{\Delta t^{inc}} \tag{A.14}$$

$\frac{\Delta t^{wc}}{\Delta t^{inc}}$ is unchanged because it is linked to the physics of the flow. Finally:

$$\frac{t^{wc}_{CPU}}{t^{inc}_{CPU}} = \frac{G}{1+p(\gamma-1)} \tag{A.15}$$

The CPU benefit is independent of α. For $\alpha = 1$, the factor is the initial one G.

Figure A.21 shows a plot of the new CPU time factor as a function of the acceleration factor γ with $p = 0.3$ and $p = 0.5$. The initial CPU time benefit is fixed in the range $5 < G < 20$ following the conclusions from the NACA test case in Section 5.

It should be noted that although the initial factor is between 5 and 20, the weakly-compressible scheme should be accelerated between 9 and 70 times to become competitive. In addition, although the code in which the weakly-compressible and the incompressible formulations have been implemented is far from optimized, the acceleration factor required is a difficult challenge. Finally, note that this short analysis does not take into account a possible acceleration of the Poisson equation, whereas it could be possible to better adjust the parameters of the PETSc and the HYPRE libraries.