European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS 2000
Barcelona, 11-14 September 2000
©ECCOMAS

CHIMERA TYPE DOMAIN DECOMPOSITION METHODS APPLIED TO FRACTIONAL STEP FINITE ELEMENT SCHEMES FOR INCOMPRESSIBLE FLOWS

M. Vázquez⋆, G. Houzeaux⋆, and R. Codina⋆

⋆International Center for Numerical Methods in Engineering (CIMNE) Universidad Politécnica de Cataluña, Jordi Girona 1-3, 08034 Barcelona, Spain, e-mail: houzeaux@cimne.upc.es, web page: http://www.cimne.upc.es/

Key words: Navier-Stokes equations, Finite Element method, CBS algorithm, Domain Decomposition method, Conservation

Abstract. *In this paper, some ideas for solving Navier-Stokes equations using fractional step techniques in combination with domain decomposition methods are introduced. Apart from its traditional use in a rational distribution of the computational work, the domain decomposition (DD) method can be a powerful tool when dealing with complex geometries, which can have moving parts. The CBS (Characteristic Based Split) general algorithm is here used to numerically solve the flow equations in each of the subdomains. It provides the required flexibility for dealing equally well with a broad range of flow regimes: incompressible and compressible, viscous and inviscid, laminar and turbulent, stationary and transient ones. As the same fluid dynamic algorithm is used for solving all these problems, the combined DD + CBS ideas inherite the fluid algorithm flexibility. A special set of transmision conditions for the subdomains and a new conservative interpolation are among the features introduced. In this work, the main ideas are set and some simple laminar incompressible problems are tackled.*

1 INTRODUCTION

The CBS general fluid mechanics algorithm was developed by O.C. Zienkiewicz and co-workers. It is widely described in many papers, among others the following: [32], [33],[8], [7] or [27]. It is a finite element algorithm, general in the sense that its twofold main purpose is to resolve the incompressibility limit and to solve viscous and inviscid problems, dealing with the numerical difficulties of each regime. The extension of fractional step techniques [5, 25] to compressible flow provides the first solution. As a bonus, equal interpolation spaces are used for pressure and velocity. On the other hand, the so called Characteristic-Galerkin numerical diffusion controls the instabilities appearing when physical diffusion is small, like in in convection dominated problems.

A Domain Decomposition method (DD) is developed to run a given problem on separated subdomains. The partition of the computational domain is performed by embedding meshes, defining a background mesh and patch meshes. This provides a great flexibility when dealing with non-trivial geometries. By "non-trivial" geometries we understand either domains containing moving parts, where a transient solution develops (for example a wing and a moving flap), or cases where stationary solutions of many different fixed geometries are sought (for instance a main sail combined with different fore and genoa sails at various positions). The classical version of such methods is known as Chimera. This work presents an alternative method using a traction (Neumann type) coupling for the velocity and a straight (Dirichlet type) coupling for the pressure on the patch mesh. This was already developed for an implicit Navier-Stokes solver in [13]. Also, we propose a "conservative" interpolation operator for the interpolation nodes.

This work is organized as follows. The physical problem is briefly described in section 2 through the set of equations that models it: the Navier-Stokes equations for incompressible flow. Section 3 introduces the CBS algorithm. How to impose different boundary conditions is there described in detail. In section 4 , the DD algorithm is introduced and its application to CBS is addressed. Section 5 presents some simple numerical examples and conclusions and future work are sketched in section 6.

2 PROBLEM DESCRIPTION

The physical problem under study in this work is the incompressible laminar flow. The so called Navier-Stokes equations of flow modell its physics. Let us picture a fluid contained in a given domain Ω and suppose density is constant in time and space. Its dynamics is then described through the vectorial field *velocity* $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t)$ and the scalar field *pressure* $p = p(\boldsymbol{x}, t)$, where $\boldsymbol{x} \in \Omega$, and time $t \in [0, \infty)$, which are in turn solution of the Navier-Stokes equations:

$$\rho \frac{\partial u_j}{\partial t} + \rho u_i \frac{\partial u_j}{\partial x_i} + \frac{\partial}{\partial x_i}(\delta_{ij} p - \tau_{ij}) = \rho g_j$$
$$\frac{\partial u_i}{\partial x_i} = 0$$

Initial and boundary conditions are imposed according to the following. For simplicity, let us take $\Gamma = \Gamma_{\mathrm{D}} \cup \Gamma_{\mathrm{N}}$. Then

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{x}, 0) &:= \boldsymbol{u}^0(\boldsymbol{x}), && \text{for all } \boldsymbol{x} \in \Omega \\ \boldsymbol{u}(\boldsymbol{x}, t) &:= \bar{\boldsymbol{u}}(\boldsymbol{x}, t), && \text{for all } \boldsymbol{x} \in \Gamma_{\mathrm{D}} \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} &:= \bar{\boldsymbol{t}}(\boldsymbol{x}, t), && \text{for all } \boldsymbol{x} \in \Gamma_{\mathrm{N}}, \end{aligned} \tag{1}$$

where $\boldsymbol{n}$ is the exterior normal versor and the *Cauchy stress tensor* $\boldsymbol{\sigma}$ is defined as

$$\sigma_{ij} = \tau_{ij} - p\,\delta_{ij}$$

being $\boldsymbol{\tau}$ the *viscous stress tensor*. Initial condition $\boldsymbol{u}^0(x_j)$ and boundary conditions $\bar{\boldsymbol{u}}(\boldsymbol{x}, t)$ and $\bar{\boldsymbol{t}}(\boldsymbol{x}, t)$ are given functions. Γ_{D} accounts for the part of the boundary with Dirichlet conditions for the velocity field, and Γ_{N}, for that with Neumann conditions for this field.

The stress tensor is related linearly to the first derivatives of the velocity, as usual in Newtonian fluids. It can be written as

$$\tau_{ij} = 2\mu s_{ij}$$

where the *strain rate* tensor is

$$s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),$$

and $\mu = \rho\nu$ is the *viscosity*, a positive constant. ρ is the (constant) density and ν the *kinematic viscosity.*

Finally, the *Reynolds number* is defined as

$$Re = UL/\nu,$$

where U and L constitute respectively a velocity module and a characteristic length, which depends on the problem geometry.

3 THE CBS ALGORITHM FOR INCOMPRESSIBLE FLOWS

As said in the introduction, the CBS algorithm is specially designed for resolving the incompressibility limit. This means that the algorithm can solve compressible problems that has regions with very low compressibility (i.e. low Mach number regions), like boundary layers. By extension, it is capable of solving pure incompressible problems, where the divergence free velocity condition is explicitly imposed, all this done using the same interpolation spaces for velocity and pressure. This is achieved by the use of *splitting* or *fractional step* techniques. In a CFD context, the concept of *splitting* was first independently introduced by G. Strang [24] and in a slightly different way by A.J. Chorin and R. Temam [5, 25]. Splitting techniques are all based in the fractional solution of the momentum equation. In the context of incompressible flows, these methods are also called *projection* methods.

The second goal is to handle the numerical instabilities yielded by the convective terms. In convection-diffusion equations, the space discretization of convective terms by the Galerkin Method can produce spurious, non-physical oscillations in the numerical solution obtained. Hence, Navier-Stokes equations are not free of this effect. A solution to this problem is proposed in the CBS method: the characteristic based time discretization, inspired by the early works [9, 21, 16]. The key relies in how time discretization is done: instead of the partial derivative, the material one is the one discretized. This is made by reformulating the continuum flow equations in a characteristics co-moving frame. If the space discretization is done then, a consistent artificial diffusion which stabilizes convective terms appears. This diffusion is similar in looks and effects to that introduced by other schemes like Streamline Upwind Petrov-Galerkin, Sub Grid Scale, etc. (see [22] or [6] for a comparison between these methods). In this case, the additional terms basically consist of the convective derivative of the spatial residual multiplied by the time step. If space is discretized using the finite element method and a characteristic based time discretization is used, we talk about a *Characteristic - Galerkin* (CG) technique. Let us briefly describe how each of these solutions works.

Characteristic - Galerkin technique. For the sake of simplicity, consider a continuum scalar field V, which evolves in time according to

$$\frac{\partial V}{\partial t} = R(V) \tag{2}$$

where the spatial residual $R(V)$ is

$$R(V) := -\, u_i \frac{\partial V}{\partial x_i} - L(V).$$

$L(V)$ is the part of a spatial residual excluding convective terms. As a matter of fact, the Navier-Stokes linear momentum equation is of this kind. Then (2) is discretized in time, according to this idea, as:

$$\Delta V = \Delta t R(V)^n - \frac{\Delta t^2}{2} u_i^n \frac{\partial R(V)^n}{\partial x_i},$$

that is: *time-discretization of transport equation* (2) *using this method has led us to conclude that temporal variation of* V *is controlled by both the residual of the equation (at first order) and its convective derivative (at second order).*

Splitting technique. The splitting, as proposed in CBS, consists of solving the linear momentum equation in two steps. Firstly, this equation is solved without the pressure gradient term. Its unknown is called *fractional momentum* (or *fractional velocity* in the context of incompressible flows): a non-physical variable, which its use endows the solving process with some particular characteristics. Then, the continuity equation is solved, using the fractional velocity. Finally, the velocity is explicitly corrected using the new pressure obtained in the previous step. In this way, the whole system solution procedure is separated in two, apart from the final explicit correction. The first step can be done fully explicitly. The resulting continuity equation is of Poisson's type, which can be solved at a low computational cost. This approach is regarded as *semi-implicit* and it is preferred here in order to expose clearly the ideas.

Let us re-arrange the Navier-Stokes equations for incompressible flow. Suppose there are no volume forces. Then

$$\begin{aligned} \rho \frac{\partial u_j}{\partial t} &= -\rho u_i \frac{\partial u_j}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_i} - \frac{\partial p}{\partial x_j} \\ \frac{\partial u_i}{\partial x_i} &= 0, \end{aligned} \tag{3}$$

are the equations to solve, together with boundary conditions (1).

After applying the Characteristic-Galerkin time discretization briefly described above, the set of equations becomes

$$\begin{aligned} \rho \frac{\Delta u_j}{\Delta t} &= -\rho u_i^n \frac{\partial u_j}{\partial x_i}^n + \frac{\partial \tau_{ij}^n}{\partial x_i} - \frac{\partial p}{\partial x_j}^{n+1} + d_j^n \\ \frac{\partial u_i^{n+1}}{\partial x_i} &= 0, \end{aligned}$$

where $\Delta u_j = u_j^{n+1} - u_j^n$, and d_j^n is the CG artificial diffusion term.

The fractional stepping, as proposed in the CBS methods, leads to the following:

$$\begin{aligned}
\rho\frac{\Delta\tilde{u}_j}{\Delta t} &= -\rho u_i^n \frac{\partial u_j}{\partial x_i}^n + \frac{\partial \tau_{ij}^n}{\partial x_i} + d_j^n, \\
\frac{\partial u_i^{n+1}}{\partial x_i} &= 0, \\
\rho\frac{\Delta u_j}{\Delta t} &= \rho\frac{\Delta\tilde{u}_j}{\Delta t} - \frac{\partial p}{\partial x_j}^{n+1}.
\end{aligned} \tag{4}$$

See how the pressure gradient is absent in the first step and the velocity is corrected in a last step. Hence, the *fractional velocity increment* can be defined as

$$\rho\Delta\tilde{u}_j = \rho\Delta u_j + \Delta t\frac{\partial p}{\partial x_j}^{n+1}. \tag{5}$$

This definition gives the key to decouple the continuity equation, because [1]

$$\frac{\partial u_i^{n+1}}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\Delta\tilde{u}_i + u_i^n - \Delta t\frac{\partial p}{\partial x_i}^{n+1}\right),$$

Then, (4) can be written as

$$\begin{aligned}
\rho\frac{\Delta\tilde{u}_j}{\Delta t} &= -\rho u_i^n \frac{\partial u_j}{\partial x_i}^n + \frac{\partial \tau_{ij}^n}{\partial x_i} + d_j^n, \\
\frac{\partial}{\partial x_i}\left(\Delta t\frac{\partial p}{\partial x_i}^{n+1}\right) &= \frac{\partial}{\partial x_i}\left(\Delta\tilde{u}_i + u_i^n\right), \\
\rho\frac{\Delta u_j}{\Delta t} &= \frac{\Delta\tilde{u}_j}{\Delta t} - \frac{\partial p}{\partial x_j}^{n+1}.
\end{aligned} \tag{6}$$

The weak form of the set (6) is needed to discretize the space using FEM. In the following, $W(\boldsymbol{x})$ and $W_j(\boldsymbol{x})$ are the usual spatial test functions of the Finite Element Method. The former is the pressure test function and the latter, the velocity (and fractional velocity)

[1]The velocity in the continuity equation can be evaluated at an intermediate time, choosing $u_i^{n+\theta} = \theta u_i^{n+1} + (1-\theta)u_i^n$ being $0 \leq \theta \leq 1$. This is explored in the cited CBS references. Here we take $\theta = 1$.

one for each of their components. Then

$$\int_\Omega W_j \rho \frac{\Delta \tilde{u}_j}{\Delta t} d\Omega = -\int_\Omega W_j \rho u_i^n \frac{\partial u_j}{\partial x_i}^n d\Omega - \int_\Omega \frac{\partial W_j}{\partial x_i} \tau_{ij}^n d\Omega + \int_\Gamma W_j n_i \tau_{ij}^n d\Gamma + D^n, \tag{7}$$

$$\int_\Omega \frac{\partial W}{\partial x_i} \Delta t \frac{\partial p}{\partial x_i}^{n+1} d\Omega = -\int_\Omega W \frac{\partial u_i^n}{\partial x_i} d\Omega + \int_\Omega \frac{\partial W}{\partial x_i} \Delta \tilde{u}_i d\Omega - \underbrace{\int_\Gamma W n_i \Delta u_i d\Gamma}_{0}, \tag{8}$$

$$\int_\Omega W_j \rho \frac{\Delta u_j}{\Delta t} d\Omega = \int_\Omega W_j \rho \frac{\Delta \tilde{u}_j}{\Delta t} d\Omega - \int_\Omega W_j \frac{\partial p}{\partial x_j}^{n+1} d\Omega, \tag{9}$$

where we have used the definition (5) in the continuity equation weak form. The boundary integral in the first equation can be explicitly evaluated, whereas that of the continuity one is neglected. This is correct for stationary cases and a boundary localized $O(\Delta t)$ approximation in transient ones. D^n is the weak form of the CG diffusion term d_j^n (for a full deduction of these equations see [32],etc.). Now the weak form can be space-discretized using the FEM method, and the discrete system obtained solved, taking into account the proper boundary conditions. Let us remark that, due to the use of CG time discretization, the Galerkin method can be now directly used, without the needing of any additional upwinding diffusion.

The final, discretized form of the Navier-Stokes equations for the CBS method, in the case of incompressible flow is

$$\begin{aligned} \boldsymbol{M} \frac{\Delta \tilde{\bar{\boldsymbol{u}}}^n}{\Delta t} &= \boldsymbol{F}_1^* - \boldsymbol{K} \bar{\boldsymbol{u}}^n, \\ \Delta t \boldsymbol{L} \bar{\boldsymbol{p}}^{n+1} &= -\boldsymbol{D} \bar{\boldsymbol{u}}^n + \boldsymbol{G}^{\mathrm{t}} \Delta \tilde{\bar{\boldsymbol{u}}}^n + \boldsymbol{F}^*, \\ \boldsymbol{M}_0 \frac{\Delta \bar{\boldsymbol{u}}_0^n}{\Delta t} &= \boldsymbol{M}_0 \frac{\Delta \tilde{\bar{\boldsymbol{u}}}_0^n}{\Delta t} - \boldsymbol{G}_0 \bar{\boldsymbol{p}}^{n+1} + \boldsymbol{F}_2. \end{aligned}$$

Vectors of nodal unknowns have been indicated by a boldface character and an overbar. Matrices $\boldsymbol{M}$, $\boldsymbol{K}$ and $\boldsymbol{G}$ are the standard mass matrix for vector fields, the matrix coming from the viscous and convective terms in the equation for the fractional velocity and the matrix coming from the gradient operator, respectively. Subscript naught in the previous equations refers to unprescribed degrees of freedom for the velocity, and $\boldsymbol{F}_2$ contains the contribution from $\Delta \tilde{\bar{\boldsymbol{u}}}^n$ and $\Delta \bar{\boldsymbol{u}}^n$ corresponding to the prescribed degrees of freedom for the latter. Matrices $\boldsymbol{D}_0$ and $\boldsymbol{G}_0^{\mathrm{t}}$ are the submatrices of $\boldsymbol{D}$ and $\boldsymbol{G}^{\mathrm{t}}$ corresponding in turn to the set of free nodes. They are related by $\boldsymbol{D}_0 = -\boldsymbol{G}_0^{\mathrm{t}}$. Vectors $\boldsymbol{F}^*$ has been introduced to take into account the boundary values of the fractional velocity. As said above, the effect of the split is that in the discretized system appears a stabilizing term for the pressure. This term makes unnecessary any compatibility condition relating the interpolation spaces for pressure and velocity. Therefore, equal interpolation spaces can be used for all the variables of the problem. Finally, as fractional velocity equation is advanced explicitly, the algorithm is not unconditionally stable, i.e. Δt must be evaluated from stability conditions [8].

3.1 Boundary conditions

How to properly implement the boundary conditions is a very important issue. When any kind of Domain Decomposition method is used, for each of the domains special care must be taken for the whole problem to be well posed. The FEM is a powerful method for implementing different boundary condition types. From now on we will speak about *Dirichlet and Neumann boundaries* (DB and NB), referring to the classical boundary types, and by *Dirichlet nodes* we will understand not only those belonging to Dirichlet boundaries but also internal nodes where the unknown is prescribed, like in the case of the so called *fringe nodes*, to be defined later.

Each of the three equations to be solved covers a different boundary condition type. The fractional velocity equation has no direct imposition on the unknown, i.e. no Dirichlet condition is used for $\tilde{u}_j$. On the other hand, Neumann boundary conditions can be (weakly) imposed through the boundary integral and expressed in terms of traction t_i, which is defined as

$$t_i = -pn_i + \tau_{ij} n_j .$$

Suppose the boundary is divided in two: $\Gamma = \Gamma_\mathrm{T} \cup \Gamma_\mathrm{F}$, where subindex "T" accounts for traction prescription and "F", for the rest. Hence, we can divide also the boundary integral:

$$\int_\Gamma W_j \tau_{ij} n_i d\Gamma = \int_{\Gamma_\mathrm{F}} W_j \tau_{ij} n_i d\Gamma + \int_{\Gamma_\mathrm{T}} W_j \left(t_j + pn_j \right) d\Gamma. \tag{10}$$

In both kinds of contour, the boundary integral can be explicitly evaluated (this subject is widely discussed in precedent works [27, 19, 33]).

In the continuity equation, a decoupled Poisson-like equation in the CBS algorithm, the boundary contribution weakly imposed is, as said above, neglected. Direct prescriptions on the pressure are placed according to the problem considered. In the final correction step for the velocity, its Dirichlet boundary conditions are imposed. In brief, on one hand, Dirichlet type conditions are imposed only in (8) and (9) upon both so called *fringe nodes*, which will be defined below, and those nodes belonging to Dirichlet boundaries. On the other hand, traction type conditions are imposed in (7).

4 THE DOMAIN DECOMPOSITION METHOD

After a brief introduction to DD, we will discuss how these ideas and CBS can be concocted together. The objective is to develop a technique capable of dealing with complex geometries and flows avoiding moving-mesh/remeshing techniques. Basically, it consists on a partition of the work domain in smaller subdomains which can be solved independently, related to each other through some boundary conditions in the interfaces. In each of the subdomains the equations can be stated considering different regimes of flow. For instance, in a high Reynolds number viscous problem, full Navier - Stokes equations are solved for the inner subdomains covering the boundary layer, while in the outer regions Euler equations (or potential flow equation) are solved. The CBS method provides its flexibility for solving all the subdomains.

4.1 Introduction

Domain decomposition methods [4, 23] have been widely used in Computational mechanics for many different purposes, including:

- The **parallelization** of the computation. The domain decomposition techniques involved in the parallelization of the computation can be algebraic [12] or geometrical [17]. They are based on the parallel solution of the matrix system, or the parallel resolution of each subdomain via multicoloring, respectively.
- The use of **different numerical strategies**: different types of elements, different order of approximations, different algebraic solvers in the case of implicit codes, etc.
- The solution of **different flow regimes**. Different physics or numerical techniques can be used to solve each one of subdomain problems. E.g, viscous/inviscid coupling [11, 1], fluid/structure interactions [3], etc.
- The **simplification of the meshing**, by assigning individual meshes for the components of the geometry. The most widely used technique is known as Chimera method [2]. It is also referred to as composite, overset or zonal grids method. Local refinement is a direct application of this concept [20].

The latter techniques can be easily combined: the use of different numerical strategies in different regions of the flow is straightforward when using composite grids. Furthermore, the algebraic solver can be parallelized on each local problem.

The DD iterative process presented in this work was introduced in [13] for incompressible flows, in the framework of an implicit finite element solver. It can be applied as a tool to simplify the mesh generation, to deal with moving subdomains, and finally, as a local refinement technique. This paper presents the extension of the method to a semi implicit fractional step finite element code. These DD methods present some basic differences due to the inherent characteristics of the finite element algorithm, and can be addressed to any of the preceding four items. In this work, particularly the fourth one is explored.

4.2 Geometrical coupling

We want to set up a simple strategy to solve a fluid problem on a given geometry, including the possibility of adding, removing and modifying easily some components, without the need for remeshing the global mesh. The DD algorithm to account for these requirements is based on a Chimera-like coupling. A background mesh is first defined. It can contain some objects which geometries and position should not change with time, and for which the grid can easily be generated. Then, separate grids are generated for the components to be patched onto the background mesh. This defines a global geometry on which the relative positions of the objects can be changed easily.

4.3 Transmission conditions

Along with the geometrical coupling of the subdmomains, some coupling variables must be carefully chosen in order to obtain a global solution from the local solutions on each subdomain. This global solution will be obtained iteratively, by exchanging pairs of variables between subdomains, according to some **transmission conditions**. The nodes involved in the transmission process are called **interface nodes**.

At this stage, we have to distinguish between transmission conditions imposed on a boundary of a subdomain and that imposed on the interior of a subdomain. On the one hand, classical **iteration-by-subdomain algorithms** like the Dirichlet/Neumann method introduced in [18], using disjoint subdomains, or the Schwarz method, using overlapping subdomains, are based on the iterative updates of the boundary conditions. The interface nodes are therefore part of the boundaries. On the other hand, the **Chimera method** consists in eliminating part of the interior of a background mesh by interpolating the solution at some nodes located inside the domains of the patch meshes. The interface nodes are interior nodes and are called **interpolation nodes**.

The importance of the distinction pointed out earlier stems from the variational formulation of the problem, for which conditions on the boundaries of the domain are required, namely the essential and the natural boundary conditions, involving the primary and the secondary variables, respectively. Therefore, the transmission conditions on boundaries are logically taken as the essential and natural boundary conditions of the weak formulation of the governing equations, given by equations (7), (8) and (9). In the framework of the present CBS method, the primary variables are the velocity and the pressure, while the secondary variables are the strain rates, or, equivalently, the traction, via equation (10).

We propose two types of couplings, namely a standard **Chimera/Dirichlet coupling** (C/D), and a **Chimera/Neumann coupling** (C/N). The background mesh is the “Chimera” subdomain while the patch mesh is either assigned Dirichlet or Neumann transmission conditions. The 3 types of transmission conditions, C holding for Chimera, D for Dirichlet, and N for Neumann are defined as follows for each coupling (see figure (1)):

- C/D coupling:
  - C: interpolation of velocity and pressure at interpolation nodes.
  - D: interpolation of velocity and prescription in equation (9), interpolation of pressure and prescription in (8) at interface nodes.
- C/N coupling:
  - C: interpolation of velocity at interpolation points.
  - N: interpolation and prescription of the pressure in (8) and the strain rate tensor in (7) at interface nodes. See section 4.6 for more details on the implementation.

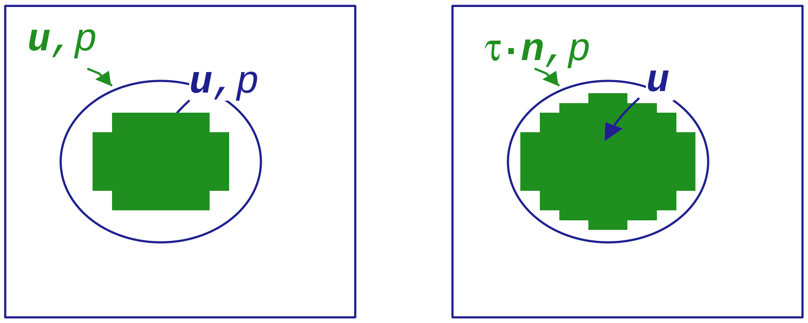


Figure 1: Two types of Chimera-type couplings. (Left) Chimera/Dirichlet. (Right) Chimera/Neumann.

"Chimera" is not actually an appropriate term to define an interface type as it generally defines a complete DD method in the scientific literature. We hope its use in the context of this paper is clear. The choices of the Chimera type transmission conditions for the C/D coupling require special care: a minimum overlap is required to avoid that nodes coincide. If this were the case, velocity and pressure would be frozen at their initial values on the coinciding nodes. This is not the case of the C/N method because the variables interpolated at the interpolation nodes are different from those interpolated at the interface nodes. We characterize the overlap in terms of element layers. The first layer is the set of elements of the patch mesh connected to its own boundary. Layer i is the set of elements connected to at least one node of the elements of layer $i-1$ and not belonging to any layer. An overlap of n_{l} layers means that the Chimera subdomain cannot have interpolation nodes with host elements belonging to layers 1 to n_{l}. This is illustrated by figure 2 (Left). The interpolation nodes (that of the background located inside the patch) located in these layers are eliminated from the interpolation process and are called overlapping nodes; recall that there is no overlapping node in the case of the C/N method. Figure 2 (Right) shows the different types of nodes participating to the DD coupling, i.e. the interpolation nodes, the interface nodes, the overlapping nodes

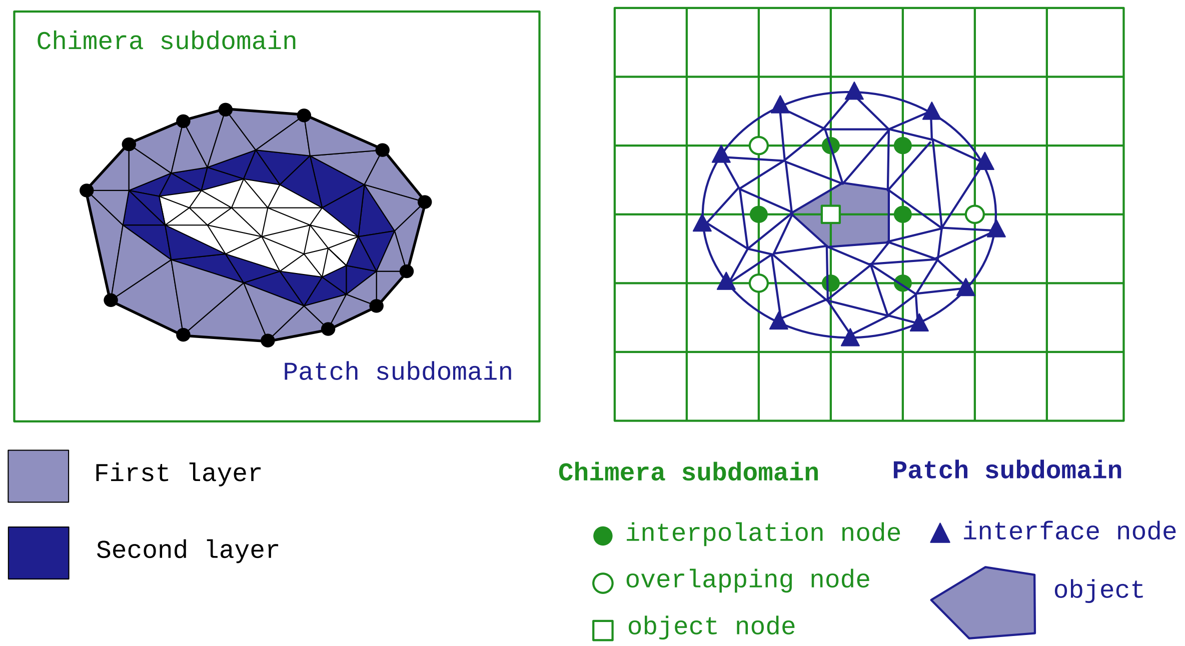


Figure 2: (Left) Definition of layers. (Right) Different types of nodes.

and, finally, the object nodes, which are the nodes of the background mesh located inside objects of the patch mesh.

Note that in the case of the C/N coupling, all the nodes of the Chimera mesh located inside the patch mesh will participate to the coupling.

4.4 Interpolation

As a first and simple approach, the transmission variables at the interpolation and interface nodes are obtained using the Lagrange interpolation functions on their corresponding host elements. This scheme is obviously not **conservative** when the meshes are of different sizes. In the case of C/D and C/N method, the background mesh is usually coarser than the patch mesh which contains the object(s) of interest. Information is therefore lost when interpolating the transmission variables on the background mesh. When Lagrange interpolation functions are used, the method will be referred to as **classical interpolation** (Class.). Let N_{p} be the total number of nodes of the patch mesh and N_{b} the total number of nodes of the background mesh. The classical method gives:

$$\mathbf{u}_{\mathrm{b}} = \mathbf{I}^{\mathrm{bp}}\mathbf{u}_{\mathrm{p}},$$

where $\mathbf{I}^{\mathrm{bp}}$ is a $N_{\mathrm{b}} \times N_{\mathrm{p}}$ matrix. Note that only the interpolation nodes are updated with this formula, and the matrix coefficients corresponding to the other nodes are meaningless.

Inspired by transfer operators of multigrid methods (see for instance [30] or [28]), we suggest an alternative method to the classical Lagrange interpolation to obtain $\mathbf{u}_{\mathrm{b}}$. We seek for a kind of transfer operator which is *conservative* in the sense that the mean of a scalar field computed from its integral in both domains is equal. Then, as in the classical interpolation, a constant field is transferred as a constant too; for identical meshes, identical fields are obtained; and, additionally, the information contained in high frequency oscillations in the fine mesh is partially present in the coarse one. Let us define the interpolation matrix $\mathbf{I}^{\mathrm{pb}}$ of the interpolation coefficients from the background mesh to the patch one. Then, analogously,

$$\mathbf{u}_{\mathrm{p}} = \mathbf{I}^{\mathrm{pb}}\mathbf{u}_{\mathrm{b}}.$$

The idea is to use the information contained in $\mathbf{I}^{\mathrm{pb}}$, which transfers variables from background to patch to improve transferring from patch to background taking into account conservation properties. This is something normally considered in multigrid methods when right hand sides are passed from finer to coarser meshes. In that case, the transpose of matrix $\mathbf{I}^{\mathrm{pb}}$ can be plainly used, no matter that the local different characteristic element sizes introduces a scale factor. It can be shown that although this scale factor is helpful in rhs' multigrid transferring, it leads to violation conservation when passing variables [29], as in the case of DD. For that reason, we propose the following

$$\mathbf{u}_{\mathrm{b}} = (\tilde{\mathbf{I}}^{\mathrm{pb}})^{t}\mathbf{u}_{\mathrm{p}}.$$

where $\tilde{\mathbf{I}}^{\mathrm{pb}}$ is the column-wise normalized interpolation matrix defined as

$$\tilde{\mathbf{I}}^{\mathrm{pb}} = \mathbf{I}^{\mathrm{pb}}\mathrm{diag}\left(1/\textstyle\sum_{i=1}^{N_{\mathrm{p}}}\mathbf{I}^{\mathrm{pb}}_{i,1}, 1/\sum_{i=1}^{N_{\mathrm{p}}}\mathbf{I}^{\mathrm{pb}}_{i,2}, \ldots, 1/\sum_{i=1}^{N_{\mathrm{p}}}\mathbf{I}^{\mathrm{pb}}_{i,N_{\mathrm{b}}}\right).$$

This method will be referred to as the **normalized transposed interpolation** (N. Tran.). To show the positive effect of normalization, we will consider momentarily also the plain transposed interpolation (Tran.). As an illustration, the three interpolations defined previously are analyzed for a very simple one-dimensional example, as shown by figure 3. Let us denote u_i, $i = 1, 2, 3, 4, 5$ the solution on the fine (patch) mesh and u_i,

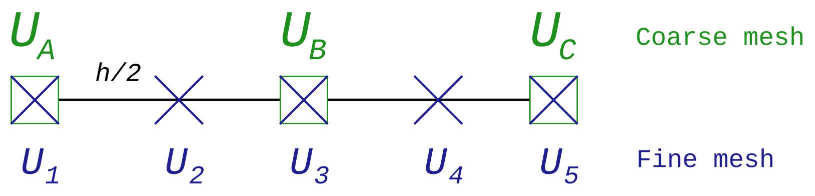


Figure 3: A simple one-dimensional example.

$i = A, B, C$ the solution on the coarse (background) mesh. The node spacing on the coarse mesh is h while that on the fine mesh mesh is twice smaller. We can obtain easily

	Fine mesh	Class.	Tran.	N. Tran.
$u_3 = 1$	$h/2$	h	h	$h/2$
$u_4 = 1$	$h/2$	0	$3h/4$	$5h/12$
$u_5 = 1$	$h/4$	$h/2$	$h/2$	$h/3$

Table 1: Integration of a triangle solution for different interpolations.

the following interpolation matrices:

$$\text{Class.:}\quad \begin{bmatrix} u_A \\ u_B \\ u_C \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix},$$

$$\text{Tran.:}\quad \begin{bmatrix} u_A \\ u_B \\ u_C \end{bmatrix} = \begin{bmatrix} 1 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix},$$

$$\text{N. Tran.:}\quad \begin{bmatrix} u_A \\ u_B \\ u_C \end{bmatrix} = \begin{bmatrix} 2/3 & 1/3 & 0 & 0 & 0 \\ 0 & 1/4 & 1/2 & 1/4 & 0 \\ 0 & 0 & 0 & 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix}.$$

The problem of the classical interpolation is obvious: the solution on the coarse mesh does not explicitly depend on u_2 and u_4. In physical terms, it means that the operator filters out the high frequencies. We will now consider some triangle solutions on the fine mesh and examine how such solutions are interpolated on the coarse mesh for the three interpolation methods described previously. Figure 4 sketches the solution obtained using last operators. Obviously, the classical method gives a continuous solution point by point. Nevertheless, it is interesting to check how well those three methods integrate the function of the interval. Consider three different triangle functions defined on the figure; table 1 shows the result of the integration of the function as calculated for each method. As expected, the Class. method gives wrong results for the integrals of the three triangles solutions. The integral is either underpredicted or overpredicted. The Tran. method always overpredict these integrals. Finally, only the N. Trans. method captures approximately the area of the three triangles solutions.

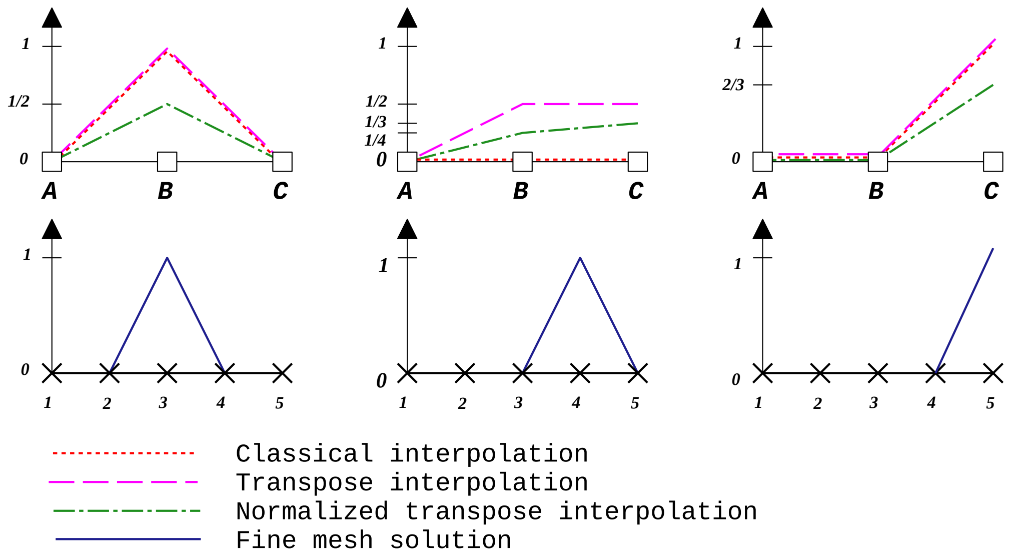


Figure 4: Interpolations of triangle solutions on the fine mesh.

4.5 The algorithm

We have defined "where" and "what" to impose as transmission conditions to set up the C/D and C/N domain decomposition methods. In the framework of an iterative process, we must now set "when". We propose 3 alternatives.

Remember that the CBS algorithm is an explicit scheme for which the time step is computed at each time iteration. There is therefore no reason why the time steps calculated by the CBS algorithm on the different subdomains should be equal. As a first approach, the time and DD iterations are coupled, i.e. transmission conditions are exchanged at each time step and each subdomain computes its own time step. This approach is logically called **coupled-asynchronous**. As a second approach, we consider a nested coupling, i.e. we enable the individual CBS algorithms work $N_{\rm o}$ time iterations before exchanging the transmission conditions with the other subdomains. This approach is called **nested-asynchronous**. These two approaches make sense when a steady state solution is wanted, i.e. when we are not interested in the solution at intermediate time steps. However, for a transient calculation the synchronization of the computations is necessary. We define therefore a **synchronous** version for which the time and DD iterations are coupled, and the same time step $\Delta t_{n_{\rm o}}$ is used for each computations at each time step $n_{\rm o}$. The time step is taken as the minimum over each subdomain i time step $\delta t(i)$ calculated at previous time iteration.

The control of the data exchange, including the transmission conditions and the time step if necessary, is performed by a Master code. Communication between the Master code and the Slave codes (solving the CBS algorithm on each subdomain) can be achieved by any of the communication librairies like PVM or MPI. The **Master/Slave strategy** performed by the Master is illustrated by Algorithm 1.

Algorithm 1 Explicit 2-domain coupling

```
set time counter $n_\text{o} = 1$
if synchronous then
  give $\Delta t_1$
  $n_\text{i} = 1$
else
  give $n_\text{i} \geq 1$
end if
while stopping criterium not reached do
  if synchronous then
    export $\Delta t_{n_\text{o}}$ to all subdomains
  end if
  export Chimera, Dirichlet and/or Neumann data to subdomain 1
  run $n_\text{i}$ time steps of subdomain 1
  import data from subdomain 1
  export Chimera, Dirichlet and/or Neumann data to subdomain 2
  run $n_\text{i}$ time steps of subdomain 2
  import data from subdomain 2
  $n_\text{o} = n_\text{o} + 1$
  if synchronous then
    $\Delta t_{n_\text{o}} = \min(\Delta t_{n_\text{o}-1}(1)), \Delta t_{n_\text{o}-1}(2))$
  end if
end while
```

The stopping criterium depends on the strategy employed. When a stationary state is wanted, it can be expressed as a condition on the norm of a certain residual. On the other hand, when a transient computation is performed, the stopping criteria is expressed as a condition on an absolute time calculated as $t = \sum_i^{n_\text{o}} \Delta t_i$. Note finally that the Chimera, Dirichlet and Neumann data can be under or over relaxed at each iteration. The effet of relaxation will be studied in next section with a numerical example.

4.6 Comments on the implementation

Conservation has been addressed for the interpolation on the Chimera subdomain, assuming the background mesh is coarser than the patch mesh. However, if the patch grid happens to be coarser than the background mesh (or at least locally coarser), information might be lost when performing the interpolation on the interface of the patch mesh. We propose two methods, although none of these is used in the numerical examples. The first one was presented in [14] and consists in constraining the interpolation by an equation for a conservation property involving the interpolated variable; e.g. the mass for the velocity, the pressure force for the pressure, etc. The second one is related to the way the strain rate continuity is treated on an interface of Neumann type. In this work, the strain rate tensor is integrated by the master program along the interface, and the data sent to the slave (patch subdomain) is the integral appearing in equation (7). If the background is locally finer than the patch mesh, integration points can be injected on the interface in order to compute the integral with higher precision and to conserve the viscous force across the interface.

Apart from its conservation property, the normalized transposed interpolation has one more advantage. The interpolation matrix involves only the interpolation coefficents of the patch mesh nodes of the corresponding host elements of the background mesh. If the background mesh is structured (Q1 elements), the search for host elements is therefore trivial. This could be an important property if the patch is moving with time; in this case, the interpolation operator would have to be calculated at each time step.

5 NUMERICAL EXAMPLES

5.1 Example 1. Exact solution

We perform an exact solution simulation on the Navier-Stokes equations with

$$\rho = 1, \mu = 1.$$

The computational domain is the unit square $[0, 1] \times [0, 1]$. The solution we want to mimic is

$$\begin{aligned} u &= x^2(1-x)^2 \mathrm{e}^x (2y - 6y^2 + 4y^3), \\ v &= -[2x - 6x^2 + 4x^3 + x^2(1-x)^2]\mathrm{e}^x y^2 (1-y)^2, \\ p &= x + y - 2, \end{aligned}$$

with homogeneous boundary condition for the velocity on the contour and imposing the pressure to zero on the top-right corner. The corresponding source term is added to the right hand side of the linear momentum equation (3).

The background mesh is the whole computational domain, while the patch mesh occupies only the bottom-left quarter of the unit square. The meshes are uniform and all nodes coincide in order to avoid possible conservation errors due to inappropriate interpolation. The reference background mesh has 100 Q1 elements and its characteristic length H is used as the unit length. The following points will be studied:

- the mesh convergence test for both the C/D and the C/N methods,
- the comparison between the synchronous method and the asynchronous method with and without inner iterations,
- the effect of the overlapping on the convergence and accuracy of the method,

- the effect of the relaxation on the convergence.

For the domain decomposition solution, we define the velocity and pressure errors, ϵ_u and ϵ_p respectively, as follows:

$$\epsilon_u = \frac{\sum_{j=1}^{2} \frac{1}{N_\mathrm{j}} \sum_{i=1}^{N_\mathrm{j}} \sqrt{(u_h(i) - u_e(i))^2 + (v_h(i) - v_e(i))^2}}{\sum_{j=1}^{2} \frac{1}{N_\mathrm{j}} \sum_{i=1}^{N_\mathrm{j}} \sqrt{u_e(i)^2 + v_e(i)^2}},$$

$$\epsilon_p = \frac{\sum_{j=1}^{2} \frac{1}{N_\mathrm{j}} \sum_{i=1}^{N_\mathrm{j}} |(p_h(i) - p_e(i)|}{\sum_{j=1}^{2} \frac{1}{N_\mathrm{j}} \sum_{i=1}^{N_\mathrm{j}} |p_e(i)|},$$

where index j holds for the subdomain number (e.g. j=1 is the background and j=2 is the patch mesh), N_j is the total number of nodes of subdomain j, subindex e refers to the exact solution while subdindex h refers to the finite element solution; finally, u is the x-component of the velocity and v the y-component. Let us finally mention that the default computational parameters, when not explicitly defined are asynchronous coupling with $n_\mathrm{i} = 1$, overlapping of length H, no relaxation. The default domain decomposition method is C/D.

Mesh convergence. The first test performed is the mesh convergence. Apart from the reference background mesh, finer background meshes are considered; they have 400, 1600 and 10000 Q1 elements and their characteristic lengths are $H/2$, $H/4$ and $H/10$, respectively. The corresponding patch meshes have therefore four times less elements. In terms of element layers, the overlapping is one layer for the coarser background and patch meshes, two layers for the second finest meshes, three layers for the third finest meshes, and finally, ten layers for the finest meshes. Figures 5 (Left, Right) show the convergence of the velocity and pressure. The C/D method and the C/N method with no overlap are compared. The C/D method gives a convergence of order between one and two for the velocity, while that of the pressure is less than one. The C/N method gives worse convergence for the velocity and the same convergence for the pressure. A close look at the solution would show that the errors concentrate around the corners. We believe that this is due to the discontinuity present in corners for the traction imposed in Neumann contours.

Synchronous *vs* asynchronous. Figure 6 (Left) show the error convergence history as a function of the CPU time of the synchronous and asynchronous versions of the DD algorithm, including the effects of performing inner iterations. The synchronous and asynchronous method with one inner iteration give the same rate of convergence although the asynchronous method gives the higher error reduction in the first time steps. No conclusion can be drawn from this last remark because this can be due to start-up effects.

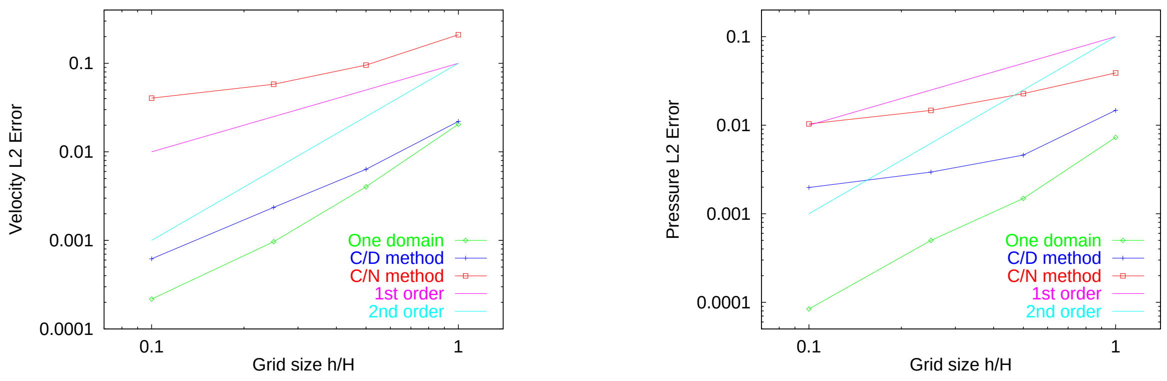


Figure 5: Example 1. (Left) Velocity errors. (Right) Pressure errors.

For $n_{\mathrm{i}} = 2$, the rate of convergence is higher than for $n_{\mathrm{i}} = 1$, and figure 6 (Right) confirms $n_{\mathrm{i}} = 2$ is the best choice. However, as shown by figure 6 (Left), the rate of convergence falls rapidly with growing number of inner iterations, and for $n_{\mathrm{i}} = 3$, the rate of convergence is already smaller than that of $n_{\mathrm{i}} = 1$.

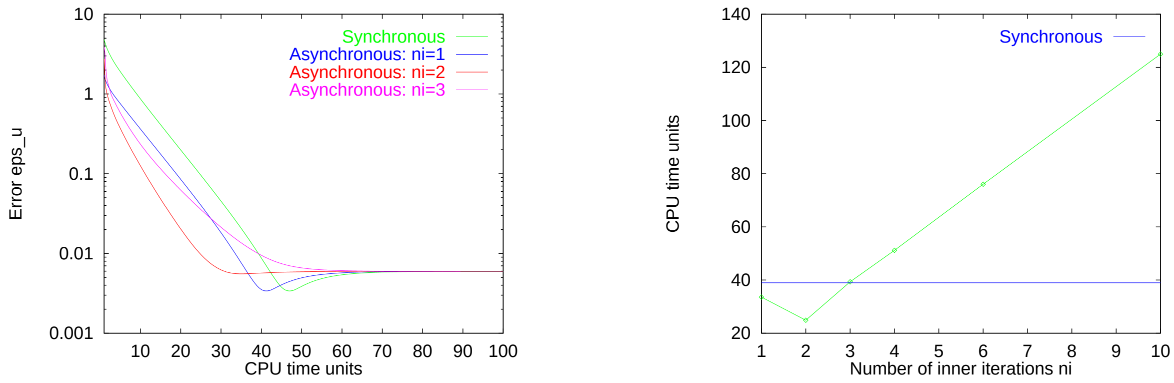
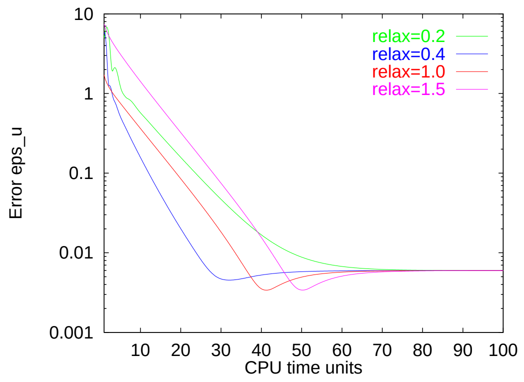


Figure 6: Example 1. (Left) Error ϵ_u for different numbers of inner iterations. (Right) Number of inner iterations to reach $\epsilon_u = 1.0 \times 10^{-2}$.

Overlapping. It is well known that for the continuous problem, as well as for finite element *implicit solvers*, the convergence of Dirichlet/Dirichlet couplings is closely related to the overlapping length (the C/D method is a kind of D/D method). The conclusion of the study on the influence of the overlapping is that the explicit C/D scheme used in this work is insensitive to the overlapping length. No figure is shown as the convergences are similar for all the overlapping lengths studied ($0.5H$,H,$1.5H$,$2H$). The convergence is therefore dominated by the explicit time advance of the fractional step technique.

Relaxation. Figures 7 (Left, Right) show the effects of the relaxation on the Chimera and Dirichlet data at each iterations. The relaxation not only helps in the first iterations but also enable to reach a faster convergence rate, as shown by figure 7 (Left). The net benefit is shown in figure 7 (Right) where it can be clearly seen that the optimum relaxation factor is around 0.4. The figure shows also that the use of relaxation requires a very fine tune around the optimum relaxation factor, which obviously, is not known a priori.



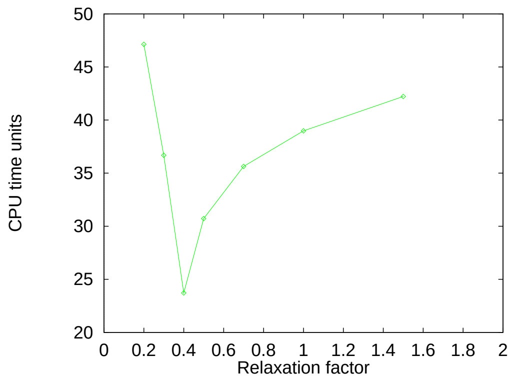


Figure 7: Example 1. (Left) Error ϵ_u for different relaxation factors. (Right) Number of CPU time units to reach $\epsilon_u = 1.0 \times 10^{-2}$.

5.2 Example 2. Vortex shedding behind a cylinder

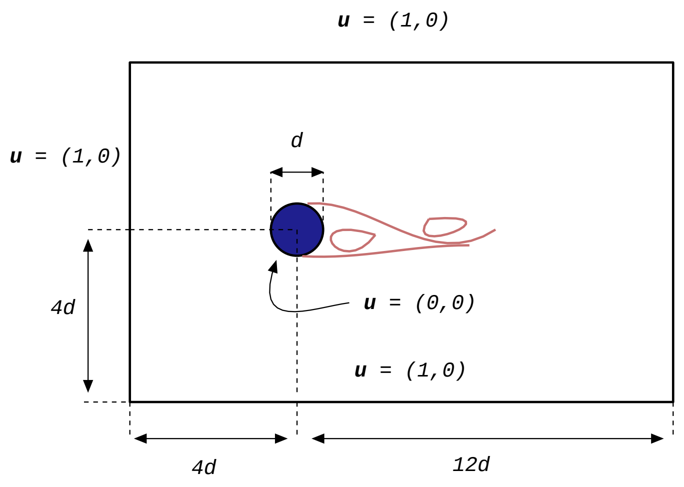


Figure 8: Example 2. Geometry, boundary conditions.

A circular cylinder is immersed in a viscous fluid. The Reynolds number Re is based on the cylinder diameter and the prescribed uniform inflow velocity. For this example, Re = 100. At this regime, if the stationary solution is perturbed, the two symmetric eddies disappear and vortex shedding occurs; the solution is therefore periodic in time.

We consider the flow bidimensional. This example will enable to analyze the behavior of the C/D and C/N methods presented in section 4.3 in a fully transient situation, as well as the two interpolation techniques, namely the Class. and N. Trans. interpolations described in section 4.4. The geometry and boundary conditions are shown in figure 8. No-slip condition is imposed on the cylinder contour. At the outflow, the velocity is free and the pressure is set to zero.

For the background mesh, two uniform Q1 meshes are considered. A fine mesh having 1800 elements and 1891 nodes (noted Fine Q1) and a coarse mesh having only 450 elements and 496 nodes (note Coarse Q1). In addition, a fine P1 background mesh was generated dividing each element of the Fine Q1 mesh into two P1 elements. This fine background mesh is noted fine P1. The patch mesh includes the cylinder and contains 3992 P1 elements and 2072 nodes (Noted P1). The overlapping for the C/D method is of one element layer. The global meshes resulting from the geometrical couplings between the Fine Q1 and P1 meshes, and that between the Coarse Q1-P1 are shown in figure 9.

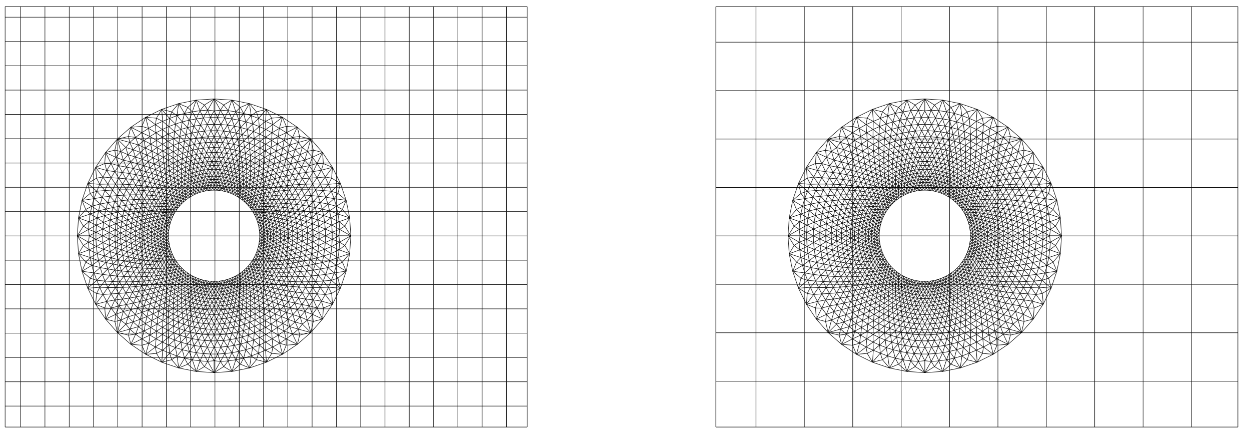


Figure 9: Example 2. Zoom on the meshes. (Left) Fine Q1 mesh. (Right) Coarse Q1 mesh.

And finally, for the sake of comparisons, the solution on one very fine P1 mesh was obtained. This mesh contains 8340 elements and 4264 nodes; the solution will be referred to as “One domain”.

Figure 10 shows the time evolution of the y-pressure force computed on the cylinder for some couplings (C/D with a one layer overlap, C/N and the Class. and N. Tran. interpolations), compared to the One domain solution on the P1 mesh. The domain decomposition results were obtained using as initial solution the velocity and pressure interpolated from the periodic solution on the One domain mesh; this explains why the results are synchronized at the first time step.

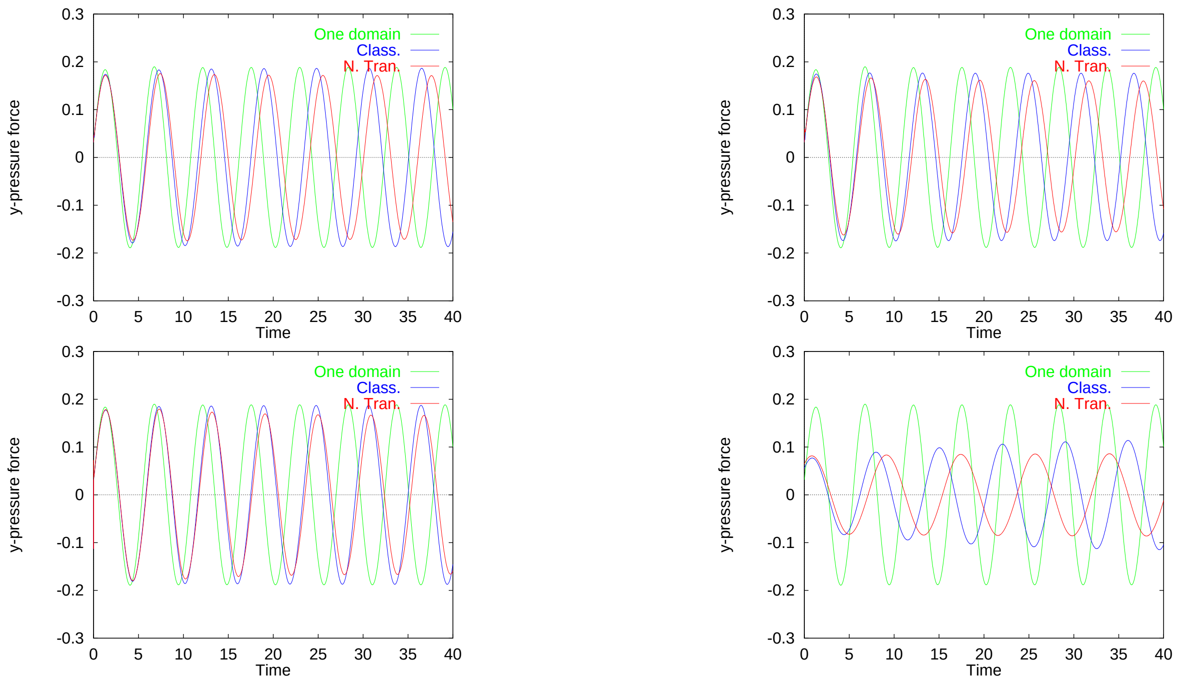


Figure 10: Example 2. Time evolution of the y-pressure force for different couplings. (Top-Left) C/D with fine Q1 and P1 meshes. (Top-Right) C/D with fine P1 and P1 meshes. (Bot.-Left) C/N with fine Q1 and P1 meshes. (Bot.-Right) C/D with coarse Q1 and P1 meshes.

The figure shows that the N. Tran. method seems more diffusive than the Class. method. This is confirmed by the results obtained for the amplitude A and period λ of the periodic y-pressure force wave, shown in tables 5.2 and 5.2: for the same mesh and same DD method, the N. Tran. interpolation gives a smaller amplitude and a higher period than the Class. interpolation. It should be pointed out that none of the method is diffusive enough to dump the oscillatory behavior with time. It can be observed finally that the C/D and C/N methods give similar results for the amplitude, while the C/N method gives a shorter period than the C/D method.

	One domain	Fine Q1-P1				Coarse Q1-P1	
	-	Class.	N. Tran.	Class.	N. Tran.	Class.	N. Tran.
λ	5.4	5.9	6.0	5.9	6.1	7.0	8.7
A	0.19	0.19	0.17	0.18	0.16	0.12	0.08

Table 2: Example 2. C/D method: A and λ computed for different meshes and interpolations.

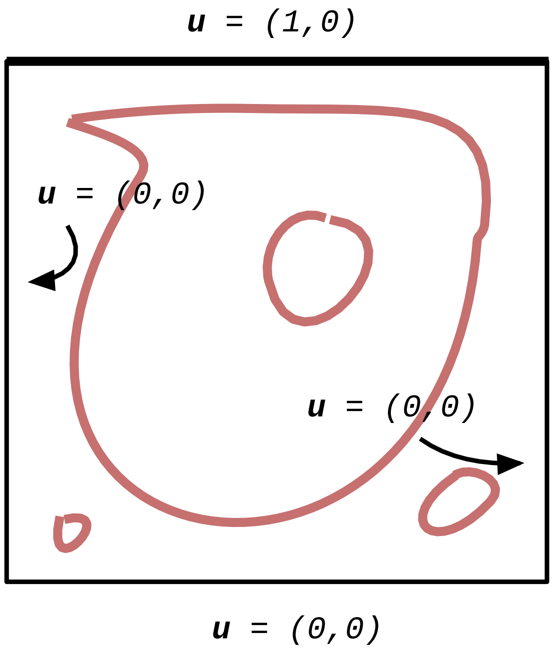
5.3 Example 3. Driven cavity flow

A viscous fluid is confined within a square cavity of unitary side, being forced to move by means of a sliding velocity in the top. No-slip condition is imposed at the bottom and

	One domain	Fine Q1-P1	
	-	Class.	N. Tran.
λ	5.4	5.8	5.9
A	0.19	0.19	0.17

Table 3: Example 2. C/N method: A and λ computed for different interpolations.

the walls; see figure 11 (Left). We consider the case Re = 5000, the characteristic length being the side length of the cavity, and the characteristic velocity being the prescribed velocity on the top wall.



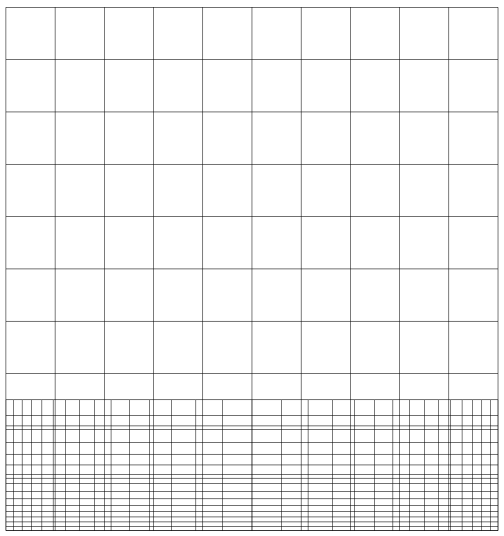


Figure 11: Example 3. (Left) Geometry and boundary conditions. (Right) Background and patch grids.

The mesh used for the background mesh occupies the whole domain and is composed of 100 uniformly distributed Q1 elements. The patch mesh has 450 elements and occupies the first quarter of the bottom part of the cavity, as shown by figure 11 (Right). The mesh is slightly refined close to the walls.

In this simple example, the conservative interpolation (N. Tran.) performance is tested. In the lower part of the cavity, the large velocity gradients present at $Re = 5000$ cannot be resolved by the very coarse background mesh. Figure 12 shows the horizontal velocity distribution along a vertical centered cut, ranging from the bottom ($y = 0$) to the center of the square. The result of the N. Tran. interpolation method is compared to that of the classical interpolation (Class.) and to that found in [10], a standard reference.

The profile obtained with the coarse grid alone is highly oscillating due to its excesively large size element, particularly in the lower vortex end, around $y = 0.05$. By using the patch grid, which can resolve smaller scales, the solution gets much closer to Ghia's results. On this basis, the DD methods of the kind proposed here can be viewed as a local refinement technique. However, if the classical interpolation is used, conservation problems can arise. On the other hand, the present method is in a better accordance with the reference results, which in turn were obtained with a highly refined mesh. In the range $0.2 < y < 0.25$, both curves obtained with the DD schemes show the overlapping region: in its outer contours the velocity values match, within the region itself the curve is double valued.

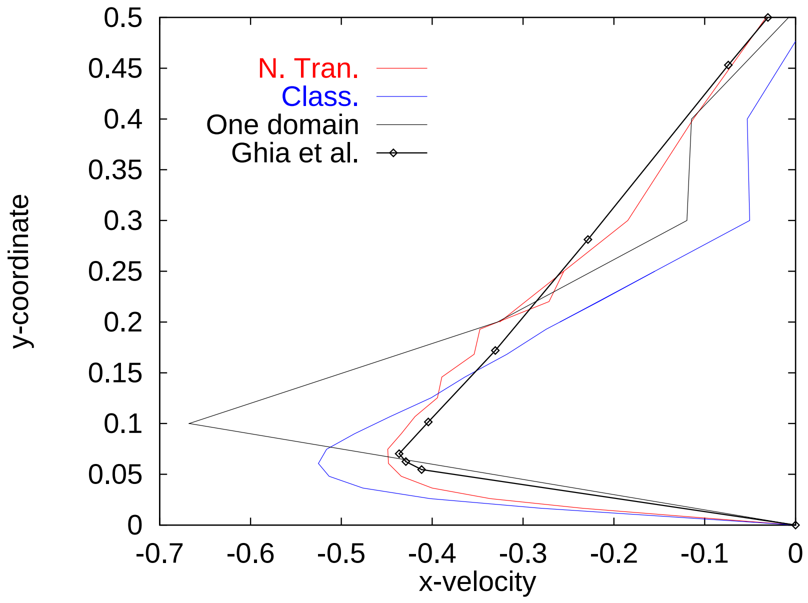


Figure 12: Example 3. Horizontal velocity along a vertical centered cut.

In figure 13, the velocity vectors and pressure level contours are shown for the DD solutions with the N.Trans. interpolation. The differences between N.Trans. and Class. approaches are too subtle to be seen with the level contours, but become apparent in figure 12. Through these figures it can be seen how coarse is the background mesh, particularly in the jagged pressure contours, and how, in spite of this fact, the patch grid corrects the solution. Figure 14 shows a close-up of the vortex formed at the left bottom corner and properly resolved by the patch.

6 CONCLUSIONS

We have presented a Chimera domain decomposition method applied to a fractional step algorithm to solve incompressible flows. In particular, two types of couplings between

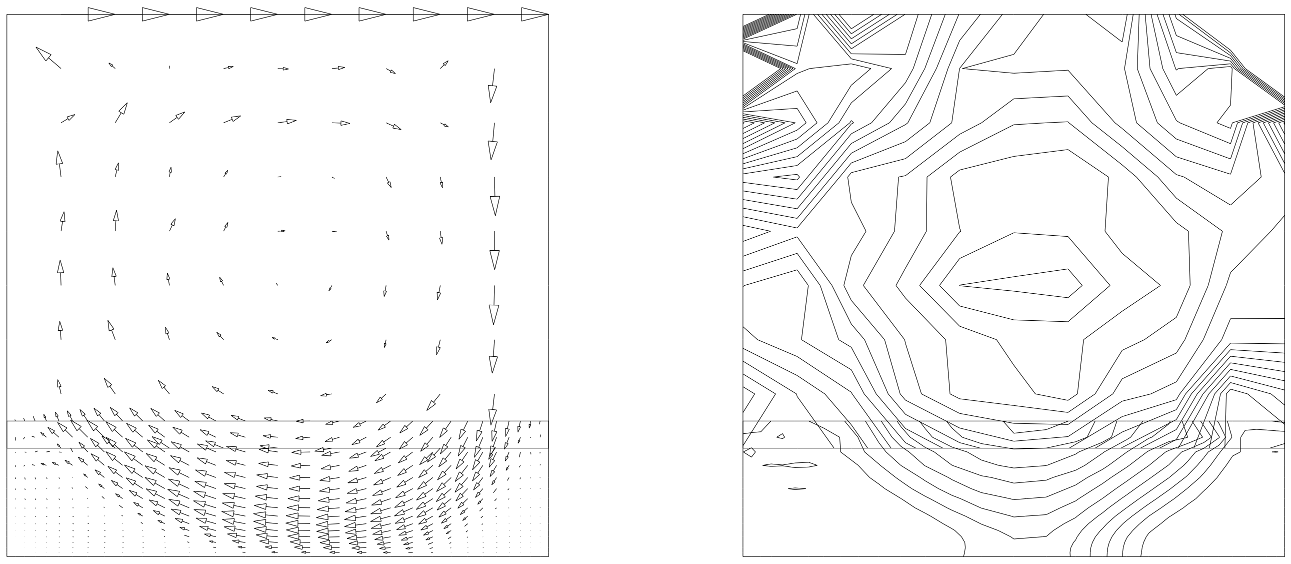


Figure 13: Example 3. (Left) Velocity vectors. (Right) Pressure contours.

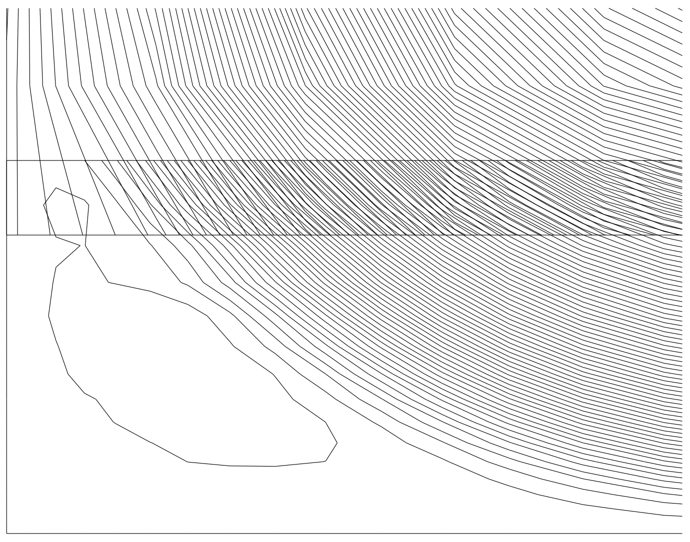


Figure 14: Example 3. Close-up of the left bottom corner vortex.

the background and patch subdomains were tested. The first one is the classical Chimera method (C/D), interpolating both velocity and pressure at the patch interface nodes, but also on the background nodes participating to the coupling. The other Chimera-like method tested (C/N) uses rather a Neumann condition on the patch interface nodes for the velocity and a Dirichlet condition for the pressure, and interpolates the velocity at the interpolation nodes of the background. The solution of an exact problem showed the difficulty of the C/N method to deal with corners. However, the C/D and C/N methods gave similar results for the solution of a fully transient flow at rather low Reynolds number, for which the interface of the patch mesh was a circle.

Two interpolation operators were tested for the interpolation of the variables on the background mesh nodes. The first one is the classical Lagrange interpolation operator; the second one is the normalized transpose of the interpolation operator of the patch mesh nodes. The study of the transient flow mentioned earlier showed that the normalized transpose operator is more diffusive than the classical operator. The last numerical example proved the efficiency of the new operator to correctly capture the solution in a critical situation at $\mathrm{Re} = 5000$, where the coupling takes part near the boundary layer.

As a future work, these methods will be applied to compressible flows. Conservation being a major issue for compressible flows, particular emphasis will be put on the study of the interpolation operator.

REFERENCES

[1] R. Arina and C. Canuto. A X–formulation of the viscous–inviscid domain decomposition for the Euler/Navier–Stokes equations. In *Domain Decomposition Methods in Scientific and Engineering Computing: Proceedings of the Seventh International Conference on Domain Decomposition*, volume 180 of *Contemporary Mathematics*, pages 453–458, Providence, Rhode Island, 1994. American Mathematical Society.

[2] J. A. Benek, P. G. Buning, and J. L. Steger. A 3–D Chimera grid embedding technique. *AIAA*, 85–1523CP, 1985.

[3] J. R. Cebral and R. Löhner. Conservative load projection and tracking for fluid-structure problems. Technical Report 96-0797, AIAA, 1996.

[4] T. F. Chan and T. P. Mathew. *Domain Decomposition Algorithms*, volume 3 of *Acta Numerica*, pages 61–143. Cambridge University Press, Cambridge, 1994.

[5] A.J. Chorin. A numerical method for solving incompressible viscous problems. *Journ. Comp. Phys.*, 2:12–26, 1967.

[6] R. Codina. Comparison of some finite element methods for solving the diffusion - convection - reaction equation. *Comp. Meth. Appl. Mech. Eng.*, 156:185–210, 1998.

[7] R. Codina, M. Vázquez, and O.C. Zienkiewicz. *A fractional step method for the solution of compressible Navier - Stokes equations* . Computational Fluid Dynamics Review. World Scientific Publishing Co., 1998. Edited by M. Hafez and K. Oshima.

[8] R. Codina, M. Vázquez, and O.C. Zienkiewicz. A general algorithm for compressible and incompressible flow–Part III. The semi-implicit form. *Int. J. Num. Meth. Fluids.*, 27:13–32, 1998.

[9] J. Douglas and T. Russel. Numerical methods for convection dominated problems based on combining the method of characteristics with finite elements or finite difference procedures. *SIAM J. Numer. Anal.*, 19:871–885, 1982.

[10] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multi-grid method. *Jour. Com. Phys.*, 48:387–441, 1982.

[11] R. Glowinsky, J. Périaux, and G. Terrasson. On the coupling of viscous and inviscid models for compressible fluid flows via domain decomposition in [26], 1990.

[12] M. Haghoo and W. Proskurowski. Parallel efficiency of a domain decomposition method. In T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund, editors, *Domain Decomposition Methods*, pages 269–281, Philadelphia, 1989. SIAM.

[13] G. Houzeaux and R. Codina. A domain decomposition method for the solution of moving subdomains in fluid dynamics. In *Proceedings of the Fourth World Congress on Computational Mechanics*, Buenos Aires (Argentina), 1998.

[14] G. Houzeaux and R. Codina. Transmission conditions with constraints in domain decomposition methods for incompressible Navier-Stokes equations. In *Proceedings of the Fourth ECCOMAS Computational Fluid Dynamics Conference*, Athens (Greece), 1998.

[15] T.J.R. Hughes. *The Finite Element Method.* Prentice-Hall, 1987.

[16] R. Loehner, K. Morgan, and O.C. Zienkiewicz. The solution of non-linear hyperbolic equations system by the finite element method. *Int. J. Num. Meth. Fluids.*, 4:1043–1063, 1984.

[17] G. Lube, A. Auge, F. C. Otto, and A. Kapurkin. Domain decomposition methods for advection-dominated problems. In *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, pages 1059–1065, Paris, France, 1996. John Wiley & Sons Ltd.

[18] L.D. Marini and A. Quarteroni. A relaxation procedure for domain decomposition methods using finite elements. *Num. Math.*, 55:575–598, 1989.

[19] T.C. Papanastasiou, N. Malamataris, and K. Ellwood. A new outflow boundary condition. *Int. J. Num. Meth. Fluids.*, 14:587–608, 1992.

[20] E. Pärt-Enander. *Overlapping Grids and Applications in Gas Dynamics.* PhD thesis, Uppsala Universitet, Sweden, 1995.

[21] O. Pironneau. On the transport-diffusion algorithm and its application to the Navier-Stokes equations. *Numer. Math.*, 38:309–332, 1982.

[22] O. Pironneau, J. Liou, and T. Tezduyar. Characteristic-Galerkin and Galerkin/least-squares space-time formulations for the advection-diffusion equation with time-dependent domains. *Comp. Meth. Appl. Mech. Eng.*, 100:117–141, 1992.

[23] A. Quarteroni. Domain decomposition and parallel processing for the numerical solution of partial differential equations. *Surv. Math. Ind.*, 1:75–118, 1991.

[24] G. Strang. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5(3):506–517, 1968.

[25] R. Temam. Sur l'approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionaires (I). *Arch. Rat. Mech. Anal.*, 32:135–153, 1969.

[26] T.F. Chan, R. Glowinski, J. Périaux, and O.B. Widlund (eds.). *Domain Decomposition methods for Partial Differential Equations*, volume 3. SIAM, Philadelphia, 1990.

[27] M. Vázquez. *Numerical Modelling of Compressible Laminar and Turbulent Flow. The Characteristic Based Split (CBS) Finite Element General Algorithm.* PhD thesis, Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports, Universitat Politècnica de Catalunya, Barcelona, 1999.

[28] M. Vázquez and R. Codina. Numerical solution of the Navier - Stokes equations using a splitting technique with Multigrid acceleration. In *Proc. 4th World Congress on Computational Mechanics, Buenos Aires, Argentina*, volume Part 2, page 663. International Asociation for Computational Mechanics, 1998.

[29] M. Vázquez, R. Codina, and O.C. Zienkiewicz. Multigrid acceleration for CBS fractional step algorithm for Navier - Stokes compressible flow equations. 2000. To appear.

[30] P. Wesseling. Introduction to Multi - Grid methods. CR - 195045 ICASE 95 - 11, NASA, 1995.

[31] O.C. Zienkiewicz. *The Finite Element Method.* Mc. Graw - Hill, London, 1977.

[32] O.C. Zienkiewicz and R. Codina. A general algorithm for compressible and incompressible flow - Part I. The Split, Characteristic-Based Scheme. *Int. J. Num. Meth. Fluids.*, 20:869–885, 1995.

[33] O.C. Zienkiewicz, K. Morgan, B.V.K. Satya Sai, R. Codina, and M. Vázquez. A general algorithm for compressible and incompressible flow - Part II. Tests on the Explicit Form. *Int. J. Num. Meth. Fluids.*, 20:887–913, 1995.