Annexe du Mémoire scientifique

présenté à

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

STÉPHANE GLOCKNER

Ingénieur de Recherche à Bordeaux-INP

pour l'obtention de

L'HABILITATION À DIRIGER DES RECHERCHES

SPÉCIALITÉ : MÉCANIQUE

Sélection des publications citées dans le mémoire

Soutenue le : 5 juin 2018

Après avis de :

MM.	Frédéric Hecht	Professeur, Université Pierre et Marie Curie, Paris	**Rapporteur**
	Mario Ricchiuto	Directeur de recherche, INRIA Bordeaux Sud-Ouest	**Rapporteur**
	Jie Shen	Professeur, Purdue University	**Rapporteur**

Devant la commission d'examen composée de :

MM.	Mejdi Azaiez	Professeur, Bordeaux-INP	**Examinateur**
	Georges-Henri Cottet	Professeur, Université Joseph Fourier, Grenoble	**Président**
	Frédéric Hecht	Professeur, Université Pierre et Marie Curie, Paris	**Rapporteur**
	Pierre Lubin	Professeur, Bordeaux-INP	**Examinateur**
	Mario Ricchiuto	Directeur de recherche, INRIA Bordeaux Sud-Ouest	**Rapporteur**
	Jie Shen	Professeur, Purdue University	**Rapporteur**

- 2018 -

Table des matières

1 A 2D block-structured mesh partitioner for accurate flow simulations on non-rectangular geometries

Computers & Fluids 43 (2011) 2–13

Contents lists available at ScienceDirect

Computers & Fluids

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

A 2D block-structured mesh partitioner for accurate flow simulations on non-rectangular geometries

E. Ahusborde *, S. Glockner

TREFLE (UMR CNRS 8508), University of Bordeaux, 33607 Pessac, France

ARTICLE INFO

Article history:
Received 3 May 2010
Received in revised form 15 July 2010
Accepted 21 July 2010
Available online 29 July 2010

Keywords:
Partitioning
Block-structured meshes
Navier–Stokes
Lid driven cavity
Non-rectangular geometries
Parallel computing

ABSTRACT

The motivation of this work is to carry out parallel simulations of incompressible flows on block-structured meshes. A new partitioning method is proposed. The quality of rectangular partitions is checked and compared with other methods, as regards load balance, edge-cut and block numbers. The partitioner is coupled with the massively parallel Hypre solver library and efficiency of the coupling is measured. Finally, the code is applied to study laminar flows (steady and unsteady) on three non-rectangular geometries. Very fine grids are used to compute reference solutions of a Z-shaped channel flow and the L-shaped and double lid driven cavities.

1. Introduction

Flow simulations on complex geometries require either block-structured or unstructured grids. The latter allow very complex geometries to be meshed leading to complex discretization schemes and solvers that require a table of connectivity between nodes and indirect addressing. If the geometry is not too complicated, it can be divided into a reasonable number of structured and conforming blocks. The volume control aspect and lexical numbering facilitate the discretization of the equations (specially if the grid remains orthogonal) and the use of the fastest parallel solvers dedicated to the structured grids.

Solver performances are closely linked to the mesh partitioning or the matrix graph. Partitioning methods can be divided into two classes: geometric and combinatorial [1]. Geometric techniques are based on the coordinates of the mesh nodes whereas combinatorial partitioning uses the graph or the hypergraph of the mesh. Geometric techniques produce lower quality partitions than combinatorial methods but are extremely fast. For unstructured meshes, partitioner libraries such as CHACO [2], METIS [3], SCOTCH [4] are available. Unfortunately, they are not well suited to the block-structured framework since they produce unstructured partitions, as shown in Fig. 1. For block-structured meshes, few works have been carried out. The two main strategies used for the partitioning of such meshes are the recursive edge bisection [5] and the so-called greedy algorithm [6]. These geometric techniques are used in elsA software [7] which is devoted to compressible flows around complex geometries. We can also cite the works of Rantakokko [8] who proposes a framework for partitioning composite grids. In our opinion, his more interesting approach consists in a graph strategy applied at the block level instead of the node level (block refinement is also proposed).

Our goal consists in providing a partitioning strategy for block-structured geometries which produces rectangular partitions. It can be classified as a geometric method even if the coordinates of the nodes are not used. The partitioner is coupled with the massively parallel solver and preconditioner Hypre library [9], more precisely with the semicoarsening geometric multigrid solver [10,11]. Firstly, we are going to present the different steps upon which the proposed method relies. Then, we will compare the quality of the partitions with other approaches and analyse the performance of the coupling with Hypre solvers. Lastly, we will apply our code to compute incompressible flows on non-rectangular geometries that have been scarcely studied so far.

2. Partitioning strategy

Firstly, let us recall the two main qualities of a partitioner:

- It must respect load balancing between processors: each processor should have nearly the same amount of work to do to minimize idle processors. In our context, each processor should have around the same number of nodes, close to the ideal load which is equal to the number of nodes divided by the number of processors.

* Corresponding author.
E-mail address: ahusborde@enscbp.fr (E. Ahusborde).

0045-7930/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2010.07.009

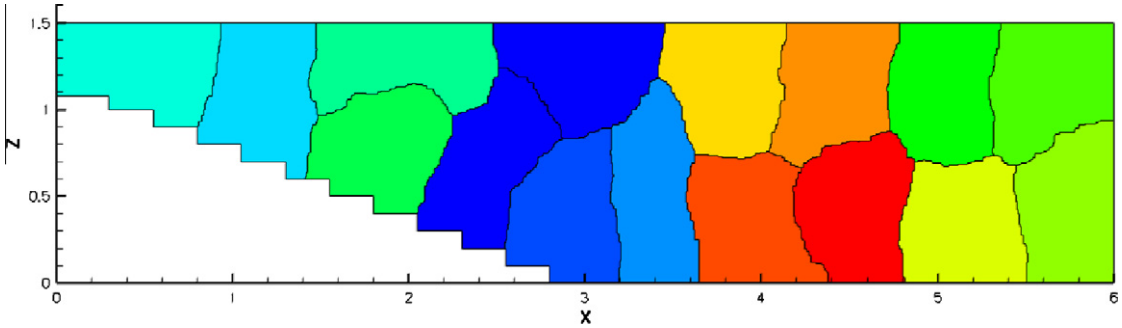


Fig. 1. Partitioning provided by METIS.

- It must minimize explicit communication between processors, *i.e.* the surface-to-volume ratio or edge-cuts. The goal is to delay as far as possible the moment when communications between processors increase such that efficiency collapses as the number of processors rises.

The conceptual interface [12] of Hypre is quite complete and supports four options: structured grid, block-structured grid, finite element interface, and linear algebraic interface. The fastest solvers such as geometric multigrid ones are available for structured grids, and block-structured grids, which is our framework. The interface requires global indexing of the nodes and rectangular boxes that can be non-contiguous. In the next sections, we will present the main steps of the partitioner and finally a complete algorithm that can be used in another solver framework.

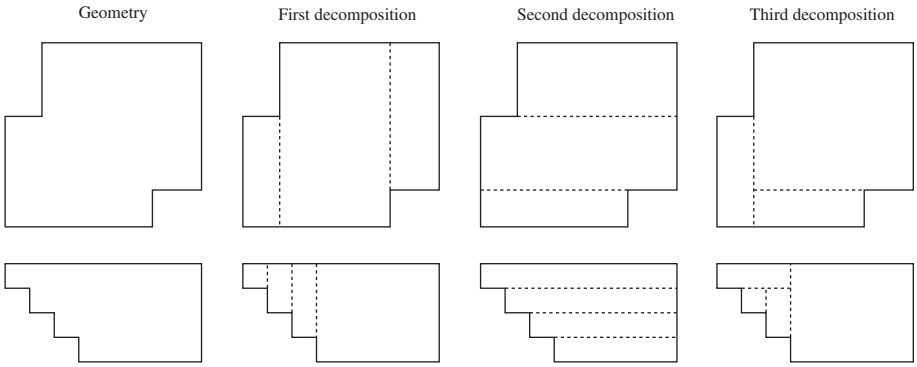


Fig. 2. Example of geometries.

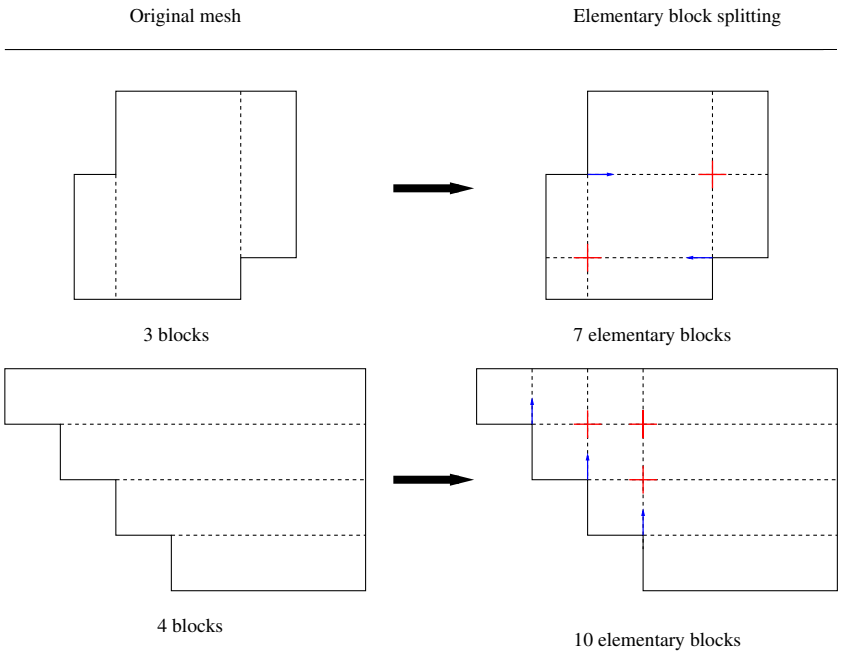


Fig. 3. Elementary block splitting.

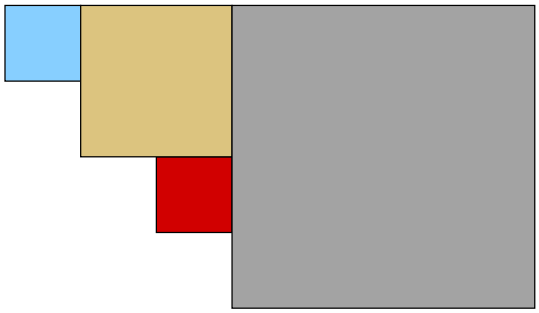


Fig. 4. Merging of elementary block into rectangular macro-blocks.

2.1. An elementary block decomposition

In our opinion, the partitioner should be independent of the initial block construction. For instance, two geometries are defined on the left part of Fig. 2. On the right part, different ways to decompose them, into three or four blocks are presented (that can be later meshed). Meshes are not shown but they are continuous through the interfaces between blocks. In the proposed method, the same partition will be produced for any geometry decomposition. This approach avoids having to consider parallelism during the construction phase of the mesh.

Consequently, the first step consists of splitting the main blocks into elementary ones. This is done by lengthening each boundary line. Intersections between lines define corners of new elementary blocks. For instance in Fig. 3, three blocks of the first geometry are split into seven elementary ones, whereas the four blocks of the second geometry are divided into 10 elementary ones. At this point, we can make three remarks:

- These elementary blocks are now the starting point of our partitioner.
- There is no reason for these blocks to be balanced.
- These blocks could have been created during the construction phase of the mesh, but it can become really fastidious for a large number of blocks.

2.2. Block merging

The second step of the partitioner consists in merging the elementary blocks into macro-blocks. In Fig. 4, we have merged 10 elementary blocks into four macro-blocks. The first macro-block is the largest of all possible macro-blocks. Then with the remaining elementary blocks, we choose the largest remaining macro-block and so on until there is no more elementary blocks. The idea of generating macro-blocks as big as possible is to minimize the number of blocks (and consequently to maximize their size) for which we are able to construct simple and optimal partitioning. Each macro-block is split into three zones (see Fig. 5). The main zone is zone 1 while zones 2 and 3 are residual zones.

2.2.1. Main zone

The size of the main zone is chosen such that it is a multiple of the ideal load. Then, straightforward partitioning that minimizes edge-cuts and respects load balancing is applied.

The number of cells in each direction of space is taken as an input by the partitioning (N_x = number of cells in the x direction, N_y = number of cells in the y direction). It can produce square or rectangular partitions. We also consider a special case if the number of processor associated to the zone 1 is a prime number (see Fig. 6).

2.2.2. Residual zones

Residual zones 2 and 3 of two different macro-blocks are associated to one processor so that the sum of their size is equal to the ideal load. Thus, load balancing is ensured (see Fig. 5).

We can note in Fig. 4 that very small macro-blocks (the darker one) can be composed just of one zone 2 or the sum of zones 2 and

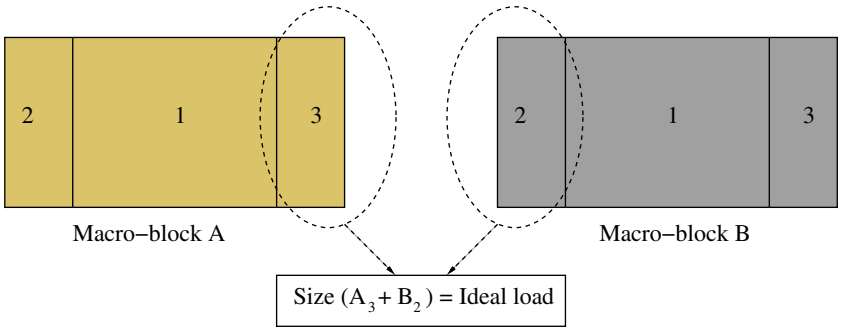


Fig. 5. Residual zones.

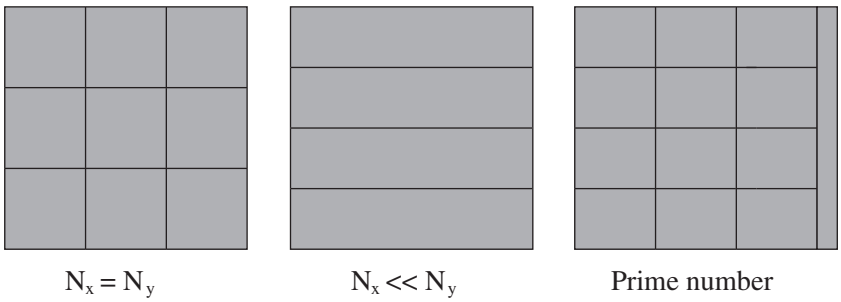


Fig. 6. Partitioning for the main zone.

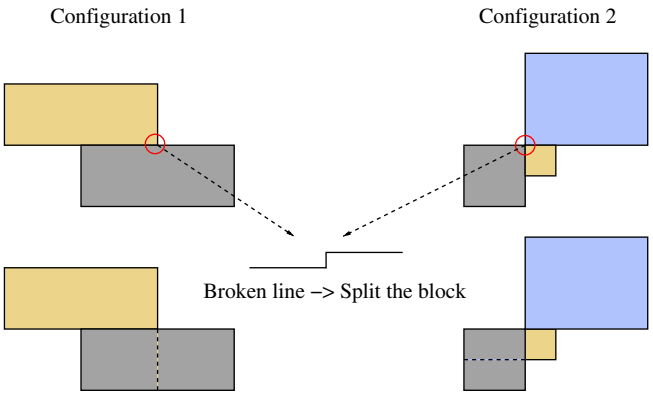


Fig. 7. Block merging rules.

3. These zones can create some non-contiguous regions on the same processor.

2.3. Node partitioning

If we had used cell partitioning, further caution would not have been necessary. With node partitioning, however, care has to be taken to the boundaries between blocks that can lead to non-rectangular macro-blocks. Indeed, as shown in Fig. 7, there exist two configurations that produce a broken boundary line. The only solution is to split the blocks in two parts. These configurations are quite rare in the cases we have studied.

From this set of geometric considerations, an Algorithm 1 has been extracted that can be applied to any 2D block-structured grid.

Algorithm 1. Partitioning strategy

1: Elementary block splitting
2: **for** Each elementary block **do**
3: Create the list of all possible rectangular macro-blocks associated to the elementary block
4: **end for**
5: **while** Macro-block partitioning not finished **do**
6: Choose the biggest remaining macro-block
7: **if** Configuration 1 **then**
8: Reject the macro-block from the list
9: **else if** Configuration 2 **then**
10: Cut the block and update macro-block list
11: **else**
12: Accept the macro-block and update macro-block list
13: **end if**
14: **end while**
15: **for** Each macro-block **do**
16: Construction of the main zone and residual zones
17: Partitioning of the main zone
18: **end for**
19: **return**

3. Block-structured partitioner quality and performance

3.1. Load balancing and edge-cuts

In this section, we test the quality of the partitioner concerning the load balancing, the number of edge-cuts, the number of blocks and the time consumed by the partitioning in comparison with those using METIS, the recursive edge bisection (REB) and the greedy algorithm (GA). For the two latters, results have been obtained with elsA software [7]. Several examples are considered.

The first example is the double cavity geometry, composed of three blocks, four elementary blocks and 4×10^5 nodes. Fig. 8 represents the partitions for 16 and 64 processors (one colour per processor). Table 1 compares load imbalance, edge-cut and block numbers between the different approaches.

The second example concerns a Z-shaped canal with three blocks, five elementary blocks and 5×10^5 nodes. Again, Fig. 9 represents the partition for 64 processors whereas Table 2 compares performances.

Table 1
Partitioner performance for the double cavity mesh.

Number of processors	8	16	32	64
Load imbalance present (%)	0.54	0.44	1.40	2.87
Load imbalance METIS (%)	1.54	2.28	3.17	3.28
Load imbalance REB (%)	17.07	17.07	17.08	17.08
Load imbalance GA 0.001 (%)	0.08	0.08	0.09	0.09
Load imbalance GA 0.05 (%)	0.97	4.49	5	4.85
Edge-cuts present	2829	4584	7513	11,360
Edge-cuts METIS	3230	4853	7507	11,176
Edge-cuts REB	2300	3800	6000	9000
Edge-cuts GA 0.001	3183	5042	7706	13,259
Edge-cuts GA 0.05	3077	4636	7303	12,961
Number of blocks present	11	19	35	66
Number of blocks METIS	8	16	32	64
Number of blocks REB	8	16	32	64
Number of blocks GA 0.001	21	49	87	171
Number of blocks GA 0.05	14	22	41	94

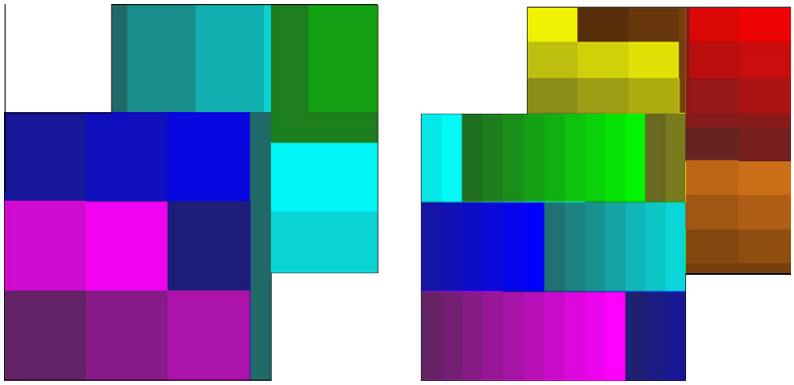


Fig. 8. Partitioning of a double cavity mesh.

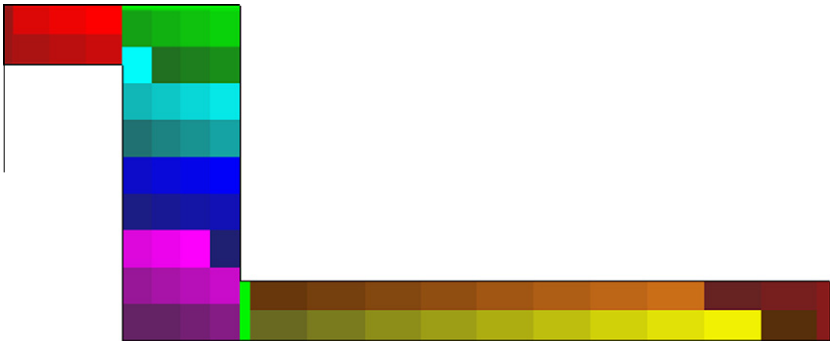


Fig. 9. Partitioning of a Z-shaped canal mesh.

Table 2
Partitioner performance for the Z-shaped canal mesh.

Number of processors	8	16	32	64
Load imbalance present (%)	0.42	0.40	0.91	1.37
Load imbalance METIS (%)	1.66	3.07	3.24	3.15
Load imbalance REB (%)	47.63	47.63	47.63	47.64
Load imbalance GA 0.001 (%)	0.08	0.09	0.08	0.1
Load imbalance GA 0.05 (%)	1.45	4.14	4.77	4.93
Edge-cuts present	2016	3692	6351	10,949
Edge-cuts METIS	2247	4055	7096	11,700
Edge-cuts REB	1773	3140	5640	8843
Edge-cuts GA 0.001	2521	4345	7625	14,317
Edge-cuts GA 0.05	2335	3864	7219	13,756
Number of blocks present	10	18	34	66
Number of blocks METIS	8	16	32	64
Number of blocks REB	8	16	32	64
Number of blocks GA 0.001	21	50	80	164
Number of blocks GA 0.05	12	20	49	91

The third and last example is a ring with 10 blocks, 27 elementary blocks and 1.5×10^5 nodes. Again, Fig. 10 represents the partition for 64 processors while Table 3 compares performances.

These test cases underline very good load imbalance lower than 1% if the number of processors is not too high. It confirms that splitting macro-blocks into three zones is efficient, residual zones being associated to verify ideal load. Lower load imbalance cannot be reached because the precision of the partitioner is equal to the length of a mesh line (necessary to keep rectangular partitions). If the number of processors increases, load imbalance increases but remains lower than 3%. Here, the partitioning effect of zone 1 is more visible: the size of a line of the mesh is relatively high in comparison with the size of the partition. The number of edge-cuts is overall very good, better than with METIS. REB method shows optimal results as regards edge-cut and block numbers but very high load imbalance (up to 47%) which is a crippling default. Moreover, this method did not provide acceptable results for the ring mesh probably because of the circular aspect of the geometry. Load imbalance produced by the GA is controlled by an epsilon parameter which has a consequence on the number of blocks generated: the lower is ϵ the lower is the load imbalance, but the greater is the number of blocks. Two values for ϵ have been used: 0.05 and 0.001. For $\epsilon = 0.001$ load imbalance is very low, edge-cut number is acceptable but the number of blocks is more than the double of the number of processors. That leads to a high number of non-contiguous subdomains associated to a processor: it can reduce solver efficiency and it increases the memory requirement due to the multiplication of ghost cells necessary to the communications between processors [7]. For $\epsilon = 0.05$, load imbalance is higher than our method, as well as edge-cut and block numbers. Finally for the ring example, CPU time shows that the proposed block-approach is much faster (nearly 30 times) than the other methods. This point could be even more relevant for 3D partitioning where CPU time is much longer. We can conclude that the proposed

Table 3
Partitioner performance for the ring mesh.

Number of processors	8	16	32	64
Load imbalance present (%)	0.10	0.15	0.35	0.86
Load imbalance METIS (%)	0.43	1.17	2.39	2.92
Load imbalance REB (%)	–	–	–	–
Load imbalance GA 0.001 (%)	0.08	0.05	0.08	0.05
Load imbalance GA 0.05 (%)	0.56	1.19	2.46	4.99
Edge-cuts present	2254	4346	7884	14,439
Edge-cuts METIS	2240	4746	10,113	16,482
Edge-cuts REB	–	–	–	–
Edge-cuts GA 0.001	12,730	14,052	15,805	19,524
Edge-cuts GA 0.05	12,504	13,409	14,800	18,342
Number of blocks present	17	25	41	73
Number of blocks METIS	8	16	32	64
Number of blocks REB	–	–	–	–
Number of blocks GA 0.001	50	68	109	180
Number of blocks GA 0.05	42	48	60	92
CPU present time (s)	0.04	0.048	0.048	0.048
CPU time METIS (s)	1.24	1.23	1.29	1.32
CPU time REB (s)	–	–	–	–
CPU time GA 0.001 (s)	1.63	1.64	1.65	1.67
CPU time GA 0.05 (s)	1.63	1.64	1.63	1.65

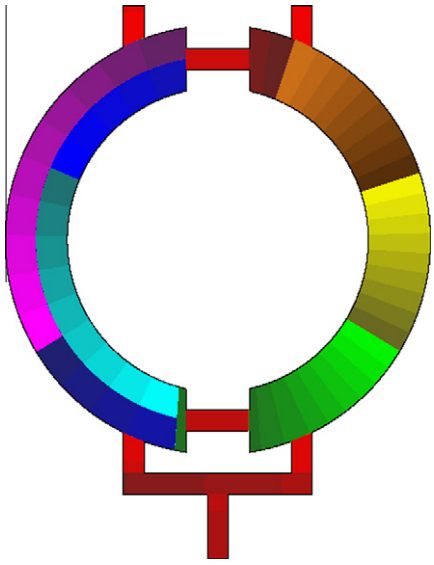


Fig. 10. Partitioning of a ring mesh.

method is efficient for the studied geometries and shows a good compromise between all the partitioning requirements.

3.2. Scalability

The efficiency of coupling the partitioner and the Hypre library is illustrated solving the Poisson equation obtained from one velocity correction step [13] which is very CPU time consuming in a Navier–Stokes solver. The studied problem is the double lid driven cavity flow.

The solver is a Generalized Minimal Residual Method (GMRES) associated to the geometric semicoarsening multigrid preconditioner. The relative residual is set to 10^{-10}. The code runs on an SGI ICE cluster. Two types of processors have been used: Harpertown nodes linked to a DDR Infiniband network and Nehalem nodes linked to a QDR Infiniband network.

In parallel computing, two types of scalability are defined. The first is the strong scaling, which represents the relation between the computation time and the number of processors for a fixed total problem size. The second is the weak scaling, for which the load per processor is constant.

3.2.1. Weak scaling

Fig. 11 displays for each type of processor (Harpertown and Nehalem) CPU time as a function of the number of processors, with 32,500 and 65,000 degrees of freedom (dof) per processor. We can see that processors Nehalem are much faster than Harpertown ones, particularly if the number of processors is low. A time ratio from 1.3 to 2.1 can be observed.

Weak efficiency is given by:

$$\mathrm{WE}\ (N) = \frac{\text{CPU time on } p \text{ processors}}{\text{CPU time on } N \text{ processors}}$$

where p denotes the number of processors used for the reference time (not always equal to one for heavy computations). Efficiency equal to one indicates an optimal behaviour for the algorithm and the computer architecture. Indeed, CPU times remains constant, equal to the reference time, while the total size of the problem increases with the number of processors. Usually, this property is hardly verified and curves with plateaus can be observed. Values of the plateaus rise toward one with the load of each processor. This phenomenon is illustrated in Fig. 12. Weak efficiency is better for the Harpertown cluster than the Nehalem one, besides a longer computation time.

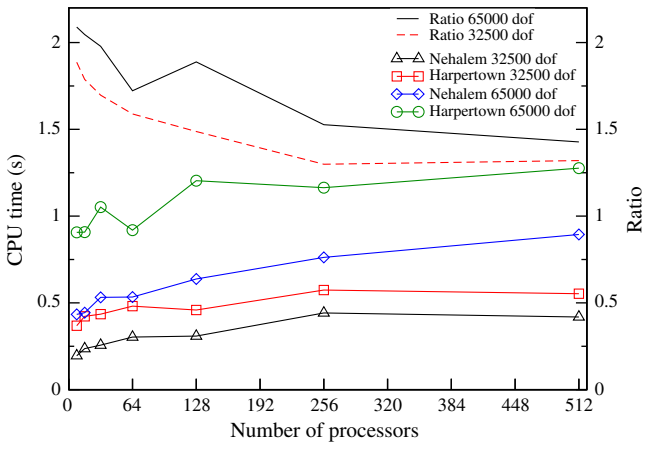


Fig. 11. CPU time versus the number of processors.

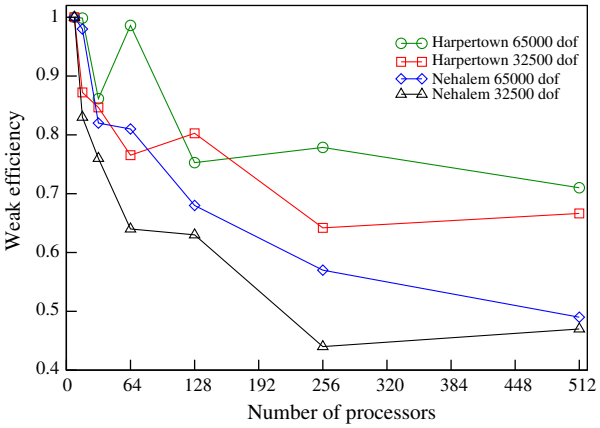


Fig. 12. Weak scaling versus the number of processors.

3.2.2. Strong scaling

Fig. 13 displays for each type of processor (Harpertown and Nehalem) on a logarithmic scale, CPU time as a function of the number of processors for two fixed size problems of 1 and 16 million degrees of freedom. Again, processors Nehalem are much faster than Harpertown ones.

Strong efficiency is given by:

$$\mathrm{SE}\ (N) = \frac{\text{CPU time on } p \text{ processors} \times p}{\text{CPU time on } N \text{ processors} \times N}$$

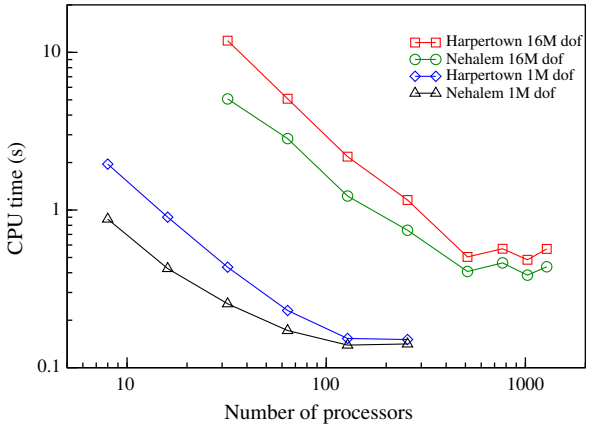


Fig. 13. CPU time versus the number of processors.

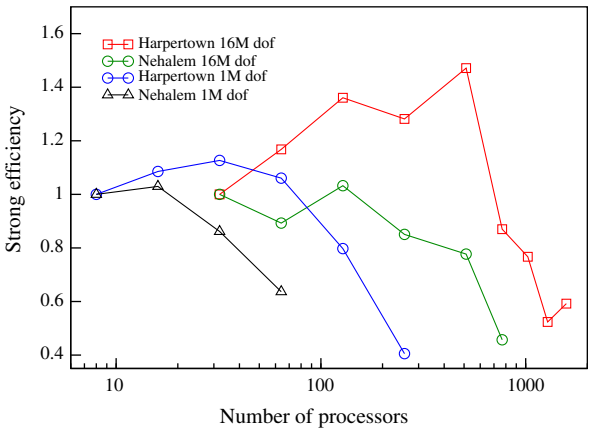


Fig. 14. Strong scaling versus the number of processors.

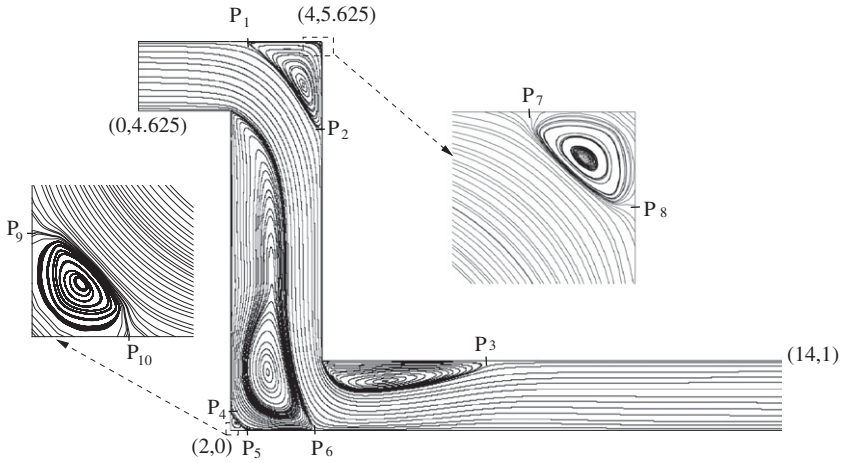


Fig. 15. Streamlines for the Z-shaped flow.

where p denotes the number of processors used for the reference time (not always equal to one for heavy computations). It points out an optimal use of the parallel resources. Efficiency equal to one indicates that communications and synchronizations between processors are negligible.

Fig. 14 represents the strong scaling versus the number of processors on a semi-logarithmic scale. With the Harpertown architecture and 16 million dof, a very high efficiency greater than 0.8 for up to 1024 processors can be observed (16,000 dof per processor). The first part of the graph being over the expected efficiency is due to memory bandwidth saturation when the number of processors is low that leads to a long reference time in the strong efficiency formula. Using more processors leads to smaller tasks that lead to a performance increase when more and more data can be kept in cache. With 1 million dof, this effect is less visible ans scaling is very good up to 128 processors (8000 dof per processor). With the Nehalem architecture which has a much higher memory bandwidth (more than three times), efficiency curves have the expected behaviour. Consequently, optimal efficiencies are obtained for 512 and 32 processors respectively for the 16 and 1 million dof problems. The saturation of the efficiency due to the increase of the communications between processors appears earlier with this architecture.

The next part of the article is devoted to the study of incompressible flows in non-rectangular geometries using the approach proposed here.

4. Computations of incompressible flows on non-rectangular geometries

Laminar flows in rectangular geometries, such as the lid driven cavity [15,16], have been extensively studied in the literature. Several numerical methods have been compared and reference solutions are available for a wide range of Reynolds numbers (leading to stationary or unsteady flows). In the present study, we propose a precise solution of flows for three non-rectangular geometries, scarcely studied so far.

Time discretization of the Navier–Stokes equations is implicit thanks to Gear's second order backward differentiation formula [14]. A pressure correction method (see Goda [13]) is used to solve the velocity–pressure coupling. Spatial discretization (second order centered scheme) is based on the finite volume method on a staggered grid of the Marker and Cells type. Solvers of the Hypre library are used.

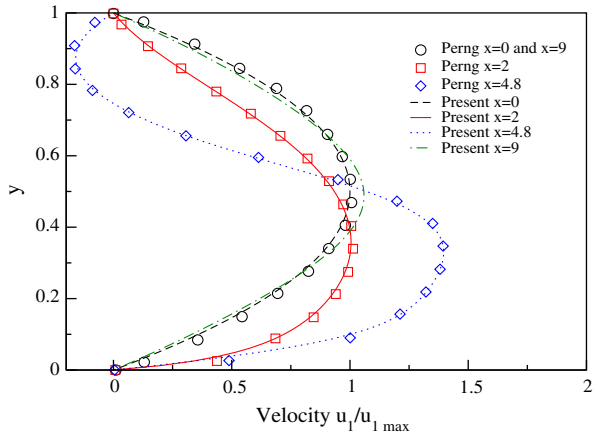


Fig. 16. Velocity u_1/u_{1max} profiles at different sections versus y.

4.1. Z-shaped channel

Flow in a Z-shaped channel has been studied in [17,18]. The channel has an abrupt expansion and contraction of it section (with a ratio of two), associated with a change of direction (horizontal contracted sections and a vertical expanded one). In our study, the Reynolds number Re is based on the width of the channel and the mean value of the inlet velocity. We can note that in [17], the Reynolds number is defined from the mean velocity u_{mean} while in [18], it is defined from the maximal velocity $u_{max} = 1.5u_{mean}$. A Poiseuille velocity profile is imposed at the inlet while Neumann and wall conditions are respectively imposed at the outlet, upper and lower sides. We have studied the flow for $Re = 200$. The streamlines and the points $P_i(x_i, y_i)$ of detachment and reattachment of the flow are presented in Fig. 15. Three main recirculations are created. The first and biggest one occupies almost one-half of the expanded region. The second eddy is developed in the upper right-hand corner while the third one is attached to the upper boundary at the beginning of the last contracted horizontal section. An infinite series of Moffat corner vortices [19] of increasingly smaller amplitude appears in the lower left and the upper right corners (see zooms in Fig. 15). Four increasingly fine grids with 3.75×10^3, 1.5×10^5, 6×10^5, 2.4×10^6 nodes were used.

4.1.1. Velocity profiles

In Fig. 16, we compare our velocity profiles at different sections with those obtained in [17]. We can note good accordance in the

Table 4
Positions (x, y) and intensities of the primary vortices.

Reference	Main primary vortex (x, y)	Vorticity
Mesh 1	(2.8097, 0.83996)	−1.29348
Mesh 2	(2.8117, 0.83881)	−1.28848
Mesh 3	(2.8127, 0.83797)	−1.28474
Mesh 4	(2.8133, 0.83748)	−1.28253
Reference	Upper primary vortex (x, y)	Vorticity
Mesh 1	(3.5884, 4.9967)	0.75678
Mesh 2	(3.5861, 4.9945)	0.75393
Mesh 3	(3.5849, 4.9925)	0.75299
Mesh 4	(3.5845, 4.9914)	0.75279
Reference	Lower primary vortex (x, y)	Vorticity
Mesh 1	(5.3985, 0.70982)	4.4441
Mesh 2	(5.3893, 0.71240)	4.3645
Mesh 3	(5.3716, 0.71392)	4.3239
Mesh 4	(5.3684, 0.71469)	4.3054

results except for the section $x = 9$ where the Poiseuille flow is restored more quickly for [17]. The very coarse grid (7000 nodes) used in [17] could explain this difference. There are no other data in the literature to compare with. We thus present our values of positions and intensities of the vortices.

4.1.2. Positions and intensities of the vortices

Table 4 represents the positions and intensities of the three primary vortices versus the mesh size for $Re = 200$. Positions and intensities of the secondary and ternary vortices are presented in Table 5. Only mesh 4 is fine enough to capture the ternary vortex. Finally, Table 6 reports the positions of the detachment and reattachment points $P_i (i = 1, 10)$ defined in Fig. 15. Convergence is achieved of two to four significant digits.

4.2. L-shaped driven cavity

Flow in an L-shaped driven cavity has been studied in [17,18,20]. The fluid is driven by the upper side while the other boundaries are walls. The configuration and the streamlines are presented in Fig. 17 for a Reynolds number $Re = 1000$ (based on the cavity width and the lid velocity). The presence of a step inside the cavity induces three recirculation zones, two in the upper part of the cavity and one in the lower. The main recirculation is located in the upper right part. Secondary vortices, whose size increases with increasing Re, appear in the corners of the cavity. If the grid is fine enough, ternary vortices can be computed. The study is carried out for $Re = 1000$. Four increasingly fine grids with 1.25×10^5, 5×10^5, 2×10^6 and 8×10^6 nodes were used.

Table 5
Positions (x, y) and intensities of the secondary vortices.

Reference	Upper secondary vortex (x, y)	Vorticity
Mesh 1	–	–
Mesh 2	(3.9676, 5.5924)	−0.00429
Mesh 3	(3.9659, 5.5907)	−0.00512
Mesh 4	(3.9655, 5.5902)	−0.00533
Reference	Lower secondary vortex (x, y)	Vorticity
Mesh 1	(2.1176, 0.10110)	0.10198
Mesh 2	(2.1188, 0.10185)	0.10283
Mesh 3	(2.1191, 0.10215)	0.10272
Mesh 4	(2.1193, 0.10219)	0.10241
Reference	Lower ternary vortex (x, y)	Vorticity
Mesh 4	(2.00632, 0.00632)	-6.22×10^{-4}

Table 6
Positions (x_i, y_i) of the detachment and reattachment points $P_i (i = 1, 10)$.

Reference	$P_1(x_1, y_1)$	$P_2(x_2, y_2)$
Mesh 1	(2.4249, 5.625)	(4, 4.3519)
Mesh 2	(2.3750, 5.625)	(4, 4.3519)
Mesh 3	(2.3500, 5.625)	(4, 4.3394)
Mesh 4	(2.3406, 5.625)	(4, 4.3332)
Reference	$P_3(x_3, y_3)$	$P_4(x_4, y_4)$
Mesh 1	(7.6250, 1)	(2, 0.20008)
Mesh 2	(7.6250, 1)	(2, 0.22499)
Mesh 3	(7.6313, 1)	(2, 0.23124)
Mesh 4	(7.6281, 1)	(2, 0.23126)
Reference	$P_5(x_5, y_5)$	$P_6(x_6, y_6)$
Mesh 1	(2.2749, 0)	(3.7750, 0)
Mesh 2	(2.2875, 0)	(3.7749, 0)
Mesh 3	(2.3063, 0)	(3.7812, 0)
Mesh 4	(2.3094, 0)	(3.7812, 0)
Reference	$P_7(x_7, y_7)$	$P_8(x_8, y_8)$
Mesh 1	–	–
Mesh 2	(3.9374, 5.625)	(4, 5.5624)
Mesh 3	(3.9249, 5.625)	(4, 5.5496)
Mesh 4	(3.9187, 5.625)	(4, 5.5437)
Reference	$P_9(x_9, y_9)$	$P_{10}(x_{10}, y_{10})$
Mesh 4	(2, 0.0125)	(2.0125,0)

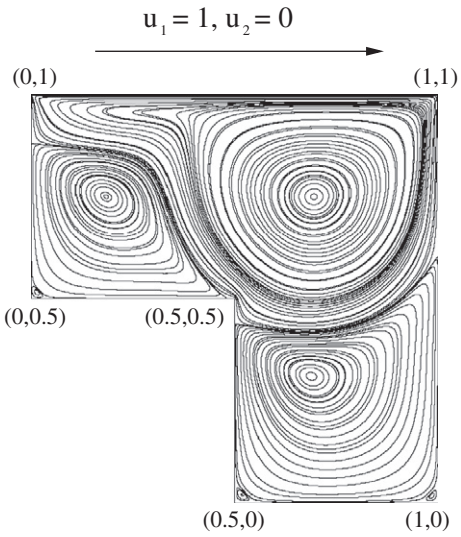


Fig. 17. Streamlines for the L-shaped driven cavity.

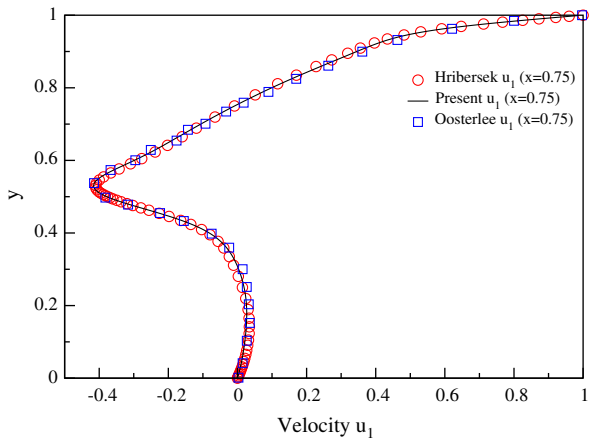


Fig. 18. u_1 profile at $x = 0.75$ versus y.

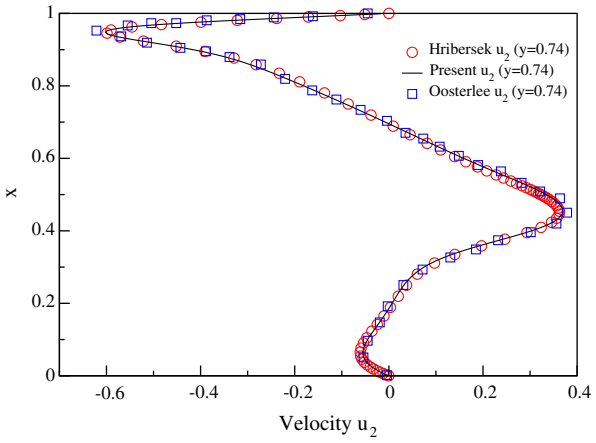


Fig. 19. u_2 profile at $y = 0.74$ versus x.

4.2.1. Velocity profiles

In Figs. 18 and 19, the velocity profiles at $x = 0.75$ and $y = 0.74$ present good accordance with those obtained in [20,18].

4.2.2. Positions and intensities of the vortices

Tables 7 and 8 represent the positions and the intensities of the primary and secondary vortices. Table 9 displays the positions and the intensities of the ternary vortices that can be only computed for meshes 3 and 4. Convergence of up to 2–4 digits is reached.

4.3. Double lid driven cavity

Flow in a double lid driven cavity has been recently studied in [21,22]. This configuration is presented in the left upper part of Fig. 20. The fluid is driven by the lower and upper boundaries in opposite directions. The other boundaries are walls. The Reynolds number Re is based on the cavity length L and the lid velocity. We focused on two flow ranges according to the value of the Reynolds number in relation to its critical value Re_c:

- $Re < Re_c$: the flow is stationary. We can note three different regimes represented in Fig. 20. On the upper right part for $Re = 100$, twin primary eddies are created between the two driving lids while secondary vortices appear in the left and right corners. In the lower left part for $Re = 1000$, the two primary vortices coalesce and two secondary vortices appear. Finally, in the lower right part for $Re = 3000$, the primary eddy becomes horizontal. The size of the two latter secondary vortices increases and two new secondary eddies appear (vertically on the upper right and lower left parts of the domain). However, it is now well-known [23] that cavity flows experience 3-dimensional global instability well below Re_c. Consequently, for $Re \geqslant Re_c^{3D}$ 2-dimensional studies are not physical anymore even if they can present numerical interests. Re_c^{3D} has been recently identified for the double and cross-sectional cavity flows [24]. For the former, it characterizes the transition between the two first flow regimes.
- $Re \geqslant Re_c$: the flow becomes 2D unsteady, from periodic to chaotic. In [15,16], a study was carried out to identify the transition from stable to periodic flow in the case of the 2D lid driven square cavity flow. The flow becomes unstable via a Hopf bifurcation. In [16], the first Lyapunov exponent was used to compute a critical Reynolds number Re_c close to 8000. In [15], thanks to a different approach (unsteady simulations with small time step), the first Hopf bifurcation occurs for $Re_c = 7402$. Several subcritical and supercritical flow regimes were identified.

Table 7
Positions (x, y) and intensities of the primary vortices.

Reference	Upper right primary vortex (x, y)	Vorticity
Oosterlee [20]	(0.6938,0.7509)	–
Mesh 1	(0.69440, 0.75113)	−3.87659
Mesh 2	(0.69408, 0.75101)	−3.87413
Mesh 3	(0.69394, 0.75097)	−3.87279
Mesh 4	(0.69388, 0.75094)	−3.87212
Reference	Upper left primary vortex (x, y)	Vorticity
Oosterlee [20]	(0.1822,0.7515)	–
Mesh 1	(0.18417, 0.75078)	1.21124
Mesh 2	(0.18386, 0.75091)	1.20968
Mesh 3	(0.18417, 0.75095)	1.20901
Mesh 4	(0.18417, 0.75097)	1.20870
Reference	Lower right primary vortex (x, y)	Vorticity
Oosterlee [20]	(0.6866,0.3089)	–
Mesh 1	(0.68741, 0.30940)	0.96079
Mesh 2	(0.68731, 0.30923)	0.96074
Mesh 3	(0.68729, 0.30918)	0.96078
Mesh 4	(0.68728, 0.30911)	0.96081

Table 8
Positions (x, y) and intensities of the secondary vortices.

Reference	Upper left secondary vortex (x, y)	Vorticity
Mesh 1	(0.01430, 0.51450)	−0.01112
Mesh 2	(0.01441, 0.51463)	−0.01140
Mesh 3	(0.01443, 0.51466)	−0.01147
Mesh 4	(0.01444, 0.51467)	−0.01148
Reference	Lower left secondary vortex (x, y)	Vorticity
Mesh 1	(0.51804, 0.01850)	−0.00780
Mesh 2	(0.51814, 0.01859)	−0.00796
Mesh 3	(0.51815, 0.01861)	−0.00801
Mesh 4	(0.51815, 0.01861)	−0.00802
Reference	Lower right secondary vortex (x, y)	Vorticity
Mesh 1	(0.98356, 0.01641)	−0.004610
Mesh 2	(0.98343, 0.01653)	−0.004739
Mesh 3	(0.98341, 0.01656)	−0.004768
Mesh 4	(0.98340, 0.01657)	−0.004778

Table 9
Positions (x, y) and intensities of the ternary vortices.

Reference	Upper left ternary vortex (x, y)	Vorticity
Mesh 3	$(6.864 \times 10^{-4}, 0.50068)$	3.17×10^{-5}
Mesh 4	$(8.279 \times 10^{-4}, 0.50082)$	7.00×10^{-5}
Reference	Lower left ternary vortex (x, y)	Vorticity
Mesh 3	$(0.99922, 7.772 \times 10^{-4})$	1.40×10^{-5}
Mesh 4	$(0.99902, 9.705 \times 10^{-4})$	3.21×10^{-5}
Reference	Lower right ternary vortex (x, y)	Vorticity
Mesh 3	$(0.50094, 9.416 \times 10^{-5})$	3.06×10^{-5}
Mesh 4	$(0.50107, 1.070 \times 10^{-4})$	5.31×10^{-5}

4.3.1. Steady flow

For the study of the steady flow, we focus on $Re = 1000$. The lower left part of Fig. 20 displays the streamlines. As described above, a primary vortex and four secondary eddies appear. With very fine meshes, ternary vortices appear between the secondary ones and the corner of the domain (see right part of Fig. 21). These vortices were not shown in previous studies [21,22]. Fig. 21 represents two zooms in which we define the points $P_i(x_i, y_i)$ of detachment and reattachment of the flow. The results found in the literature are given on the intensities and positions of the vortices.

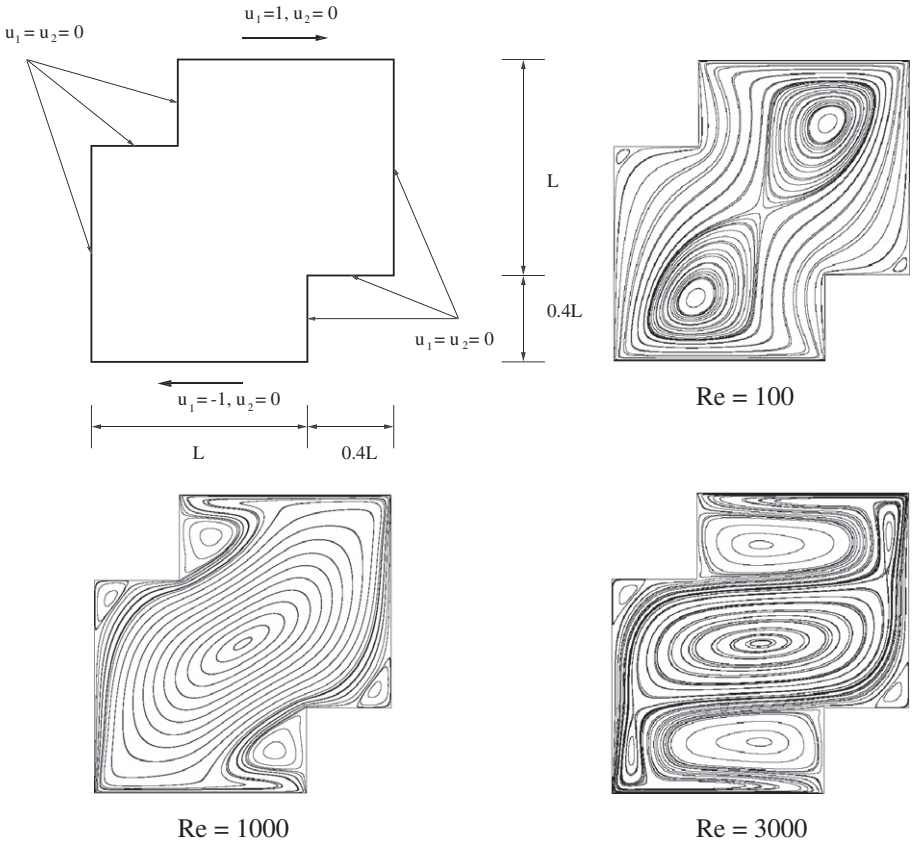


Fig. 20. Configuration of the double lid driven cavity and streamlines for different Reynolds numbers.

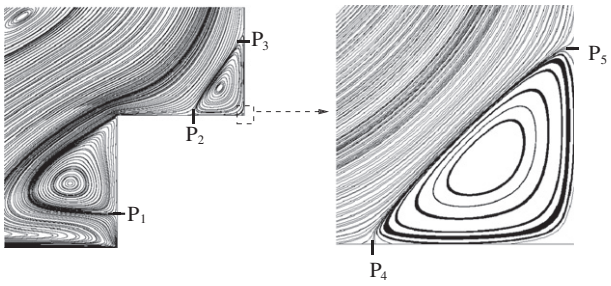


Fig. 21. Streamlines for the double lid driven cavity for $Re = 1000$.

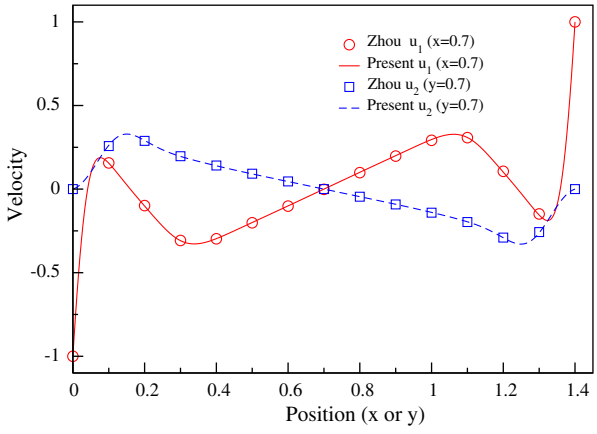


Fig. 22. Velocity profiles at $x = 0.7$ and $y = 0.7$.

In the present study, the results were obtained with convergence criteria on stationarity below 10^{-12} between two consecutive iterations. Four increasingly fine grids with 6.25×10^4, 2.5×10^5, 10^6 and 4×10^6 nodes have been used.

Table 10
Positions (x, y) and intensities of the main vortex.

Reference	Main vortex (x, y)	Vorticity
Zhou [21]	(0.70000, 0.70000)	−1.41562
Nithiarasu [22]	(0.68950, 0.69690)	−1.52363
Present (mesh 1)	(0.70099, 0.70070)	−1.49753
Present (mesh 2)	(0.69996, 0.69996)	−1.49974
Present (mesh 3)	(0.70000, 0.70000)	−1.49858
Present (mesh 4)	(0.69999, 0.69999)	−1.49838

Table 11
Positions (x, y) and intensities of the secondary vortices.

Reference	Lower secondary vortex (x, y)	Vorticity
Zhou [21]	(0.72560, 0.20000)	2.38559
Nithiarasu [22]	(0.85230, 0.20150)	2.60588
Present (mesh 1)	(0.84952, 0.19744)	2.60184
Present (mesh 2)	(0.85055, 0.19637)	2.59861
Present (mesh 3)	(0.85112, 0.19587)	2.59770
Present (mesh 4)	(0.85139, 0.19563)	2.59745
Reference	Right secondary vortex (x, y)	Vorticity
Zhou [21]	(1.32500, 0.48440)	0.53846
Nithiarasu [22]	(1.32210, 0.48360)	0.65005
Present (Mesh 1)	(1.32226, 0.48356)	0.63099
Present (Mesh 2)	(1.32243, 0.48353)	0.62956
Present (Mesh 3)	(1.32249, 0.48349)	0.62971
Present (Mesh 4)	(1.32253, 0.48346)	0.62996

Table 12
Positions (x, y) and intensities of the right ternary vortex.

Reference	(x, y)	Vorticity
Mesh 1	–	–
Mesh 2	(1.39579, 0.40420)	−0.002911
Mesh 3	(1.39531, 0.40468)	−0.004380
Mesh 4	(1.39522, 0.40478)	−0.004718

Table 13
Positions (x_i, y_i) of the detachment and reattachment points $P_i (i = 1,5)$.

Reference	$P_1(x_1, y_1)$	$P_2(x_2, y_2)$	$P_3(x_3, y_3)$
Mesh 1	(1, 0.09999)	(1.23999, 0.4)	(1, 0.63999)
Mesh 2	(1, 0.09999)	(1.23748, 0.4)	(1, 0.64499)
Mesh 3	(1, 0.09874)	(1.23624, 0.4)	(1, 0.64874)
Mesh 4	(1, 0.09937)	(1.23563, 0.4)	(1, 0.64999)
Reference	$P_4(x_4, y_4)$		$P_5(x_5, y_5)$
Mesh 1	–		–
Mesh 2	(1.3925, 0.4)		(1, 0.4075)
Mesh 3	(1.3900, 0.4)		(1, 0.4100)
Mesh 4	(1.3887, 0.4)		(1, 0.4112)

4.3.1.1. Velocity profiles. In Fig. 22, the velocity profiles at $x = 0.7$ and $y = 0.7$ present good accordance with those obtained by Zhou et al. [21].

4.3.1.2. Positions and intensities of the vortices. Tables 10 and 11 represent the positions and intensities of the primary and secondary vortices as a function of the mesh size for $Re = 1000$. Positions and intensities of the ternary vortex are shown in Table 12. For the sake of conciseness, we only focus on the positions and the intensities of one ternary and two secondary vortices. The others can be obtained symmetrically in relation to the center of the cavity. We observed symmetrical values to up to four or five significant digits. Table 13 displays the positions of the detachment and reattachment points $P_i (i = 1,5)$ defined in Fig. 21. In [21,22], the authors use coarser grids which could explain the difference between their values and ours for the primary vortices.

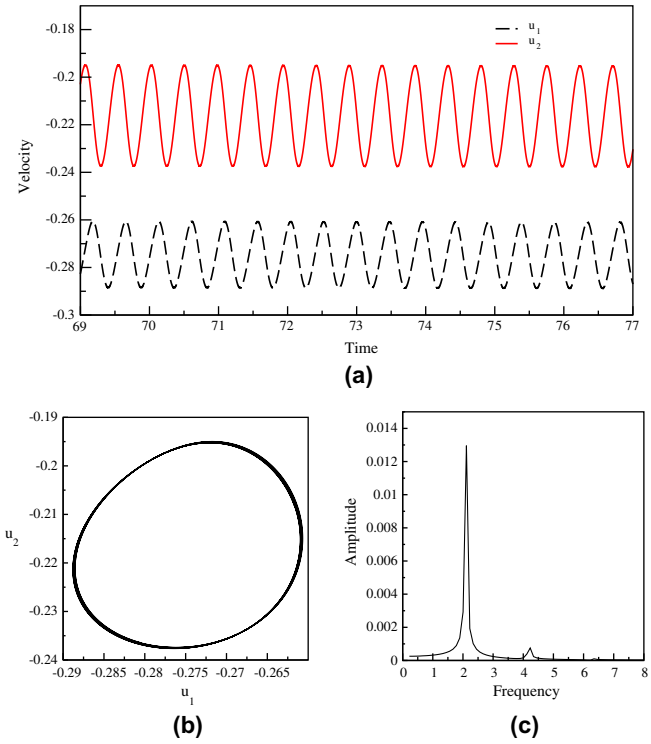


Fig. 23. Flow for $Re = 5000$. (a) u_1 and u_2 versus time t. (b) Phase trajectory on $u_1 - u_2$ plane. (c) Fourier power spectrum of the u_1 velocity.

4.3.2. Unsteady flow

In this section, we propose a first insight of an unsteady double lid driven cavity. The transition process is illustrated for $Re = 5000$ and $Re = 10{,}000$ using time velocity histories, Fourier power spectra and phase-space trajectories.

In the present case, we used mesh 3 (10^6 nodes). Flow becomes unsteady and periodic for $3480 < Re < 3500$ whereas Zhou [21] presents an unsteady flow at $Re = 3200$. This difference can be due to his coarser grids.

For $Re = 5000$, the time evolution of u_1 and u_2 variables, the phase trajectory on $u_1 - u_2$ plane and the power spectrum at the point $(x = 1, y = 1)$ between $t = 69$ s and $t = 77$ s are represented in Fig. 23. The flow is periodic with a fundamental frequency equal to 2.1151 Hz. Fig. 24 represents the evolution of the streamlines during a period giving prominence to the periodicity of the flow for $Re = 5000$. The mean flow is not symmetrical. The right lower vortex is very unstable while the others remain quite stable. This phenomenon has already been noticed in the 2D lid driven square cavity [16].

For $Re = 10{,}000$, time evolution of u_1 and u_2 variables, the phase trajectory on u_1–u_2 plane and the power spectrum at the point

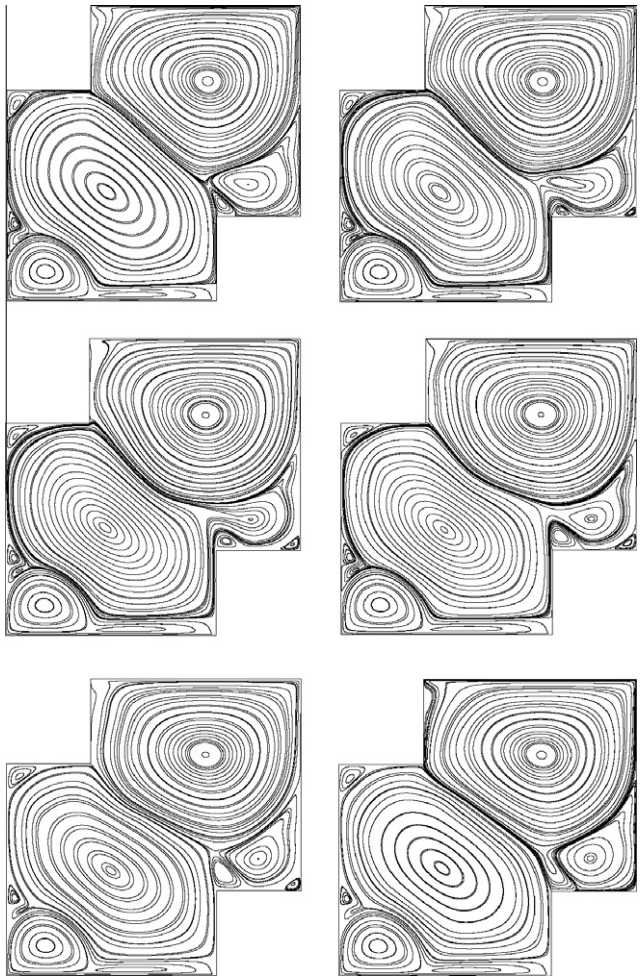


Fig. 24. Evolution of the streamlines during a period at $Re = 5000$ (it reads from left to right and vertically).

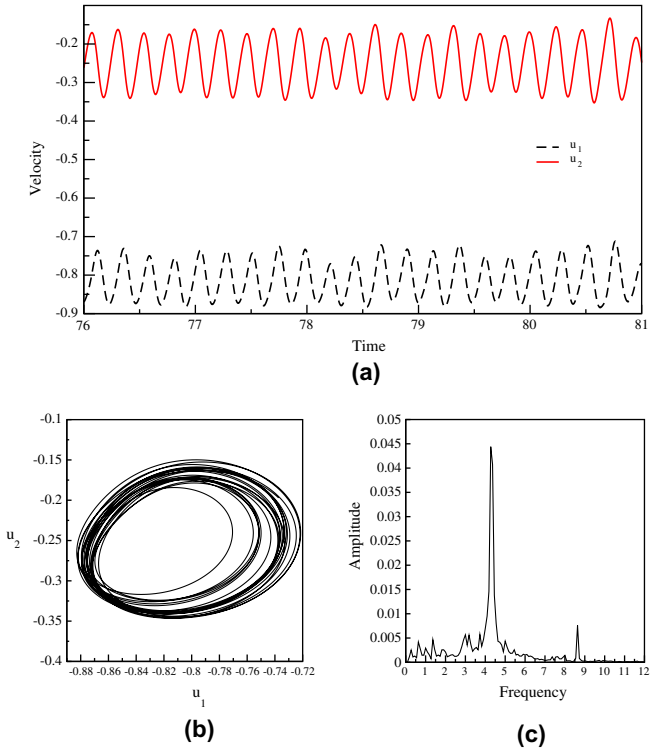


Fig. 25. Flow for $Re = 10000$. (a) u_1 and u_2 versus time t. (b) Phase trajectory on $u_1 - u_2$ plane. (c) Fourier power spectrum of the u_1 velocity.

$(x = 1, y = 1)$ between $t = 76$ s and $t = 81$ s are represented in Fig. 25. The flow becomes quasiperiodic with non-negligible variations in amplitude. The main frequency equals 4.2729 Hz.

5. Conclusion

In this paper, we proposed a method for partitioning 2D block-structured meshes. The goal was to compute flow simulations on non-rectangular geometries. Our geometrical partitioner was coupled with the massively parallel solver and preconditioner Hypre library. Several examples of partitioning are presented to check, both the efficiency and the performance of our strategy in comparison with other partitioners. Finally, we computed flow on non-rectangular geometries with very fine grids to propose reference solutions.

Acknowledgements

The authors thank Professor Michel Deville for his invitation to contribute to this special issue and Michel Gazaix, Engineer at ONERA who provided the partitioning results for the REB and GA methods. We acknowledge the calculation facilities financially supported by the Conseil Régional d'Aquitaine and the French Ministry of Research and Technology.

References

[1] Dongarra J, Foster I, Fox G, Gropp W, Kennedy K, Torczon L, et al. Sourcebook of parallel computing. Morgan Kaufmann Publishers Inc.; 2003.
[2] Hendrickson B, Leland R. The Chaco users guide, version 2.0. Technical report, Sandia National Laboratories, Albuquerque, NM; July 1995.
[3] Karypis G, Kumar V. METIS: unstructured graph partitioning and sparse matrix ordering system, Technical report, University of Minnesota, Computer Science Department, Minneapolis; 1995.
[4] Pellegrini F. SCOTCH 3.1 users guide. Technical report 1137-96, LaBRI, University of Bordeaux, France; 1996.
[5] Berger MJ, Bokhari SH. A partitioning strategy for nonuniform problems on multiprocessors. IEEE Trans Comput 1987;C-36:571–80.
[6] Ytterström A. A tool for partitioning structured multiblock meshes for parallel computational mechanics. Int J Supercomput Appl High Perform Comput 1997;11:336–43.
[7] Gourdain N, Gicquel L, Montagnac M, Vermorel O, Gazaix M, Staffelbach G, et al. High performance parallel computing of flows in complex geometries: I. Methods. Comput Sci Discovery 2009;2 [art. no. 015003].
[8] Rantakokko J. Partitioning strategies for structured multiblock grids. Parallel Comput 2000;26(12):1661–80.
[9] Falgout RD. Hypre high performance preconditioners user's manual. Center for Applied Scientific Computing, Lawrence Livermore National Laboratory; 2008.
[10] Schaffer S. A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients. SIAM J Sci Comput 1998;20:228–42.
[11] Brown PN, Falgout RD, Jones JE. Semicoarsening multigrid on distributed memory machines. SIAM J Sci Comput 2000;21:1823–34.
[12] Falgout RD, Jones JE, Yang UM. Conceptual interfaces in hypre. Future Generation Comput Sys 2006;22:239–51 [Special issue on PDF software].
[13] Goda K. A multistep technique with implicit difference schemes for calculating two and three dimensional cavity flows. J Comput Phys 1979;30:76–95.
[14] Gear C. Numerical initial value problems in ordinary differential equation. Prentice-Hall; 1971.
[15] Peng YF, Shiau YH, Hwang RR. Transition in a 2-D lid driven cavity flow. Comput Fluids 2003;32:337–52.
[16] Bruneau CH, Saad M. The 2D lid-driven cavity problem revisited. Comput Fluids 2006;35:326–48.
[17] Perng CY, Street L. A coupled multigrid-domain-splitting technique for simulating incompressible flows in geometrically complex domains. Int J Numer Methods Fluids 1991;13:269–86.
[18] Hribersek L, Skerget L. Boundary domain integral method for high Reynolds viscous fluid flows in complex planar geometries. Comput Methods Appl Mech Eng 2005;194:4196–220.
[19] Moffatt HK. Viscous and resistive eddies near a sharp corner. J Fluid Mech 1964;18:1–18.
[20] Oosterlee CW, Wesseling P, Segal A, Brakkee E. Benchmark solutions for the incompressible Navier–Stokes equations in general co-ordinates on staggered grids. Int J Numer Methods Fluids 1993;17:301–21.
[21] Zhou YC, Patnaik BSV, Wan DC, Wei GW. DSC solution for flow in a staggered double lid driven cavity. Int J Numer Methods Fluids 2003;57:211–34.
[22] Nithiarasu P, Liu CB. Steady and unsteady incompressible flow in a double driven cavity using the artificial compressibility (AC)-based characteristic-based split (CBS) scheme. Int J Numer Methods Fluids 2005;63:380–97.
[23] Theofilis V. Advances in global linear instability analysis of nonparallel and three-dimensional flows. Prog Aerospace Sci 2003;39:249–315.
[24] De Vicente J, Rodríguez D, Theofilis V, Valero E. On high-Re numerical solutions in spanwise-periodic lid-driven cavity flows with complex cross-sectional profiles. Comput Fluids, submitted for publication.

2 Resolution of the Navier–Stokes equations on block-structured meshes

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **54**:1239–1268
Published online 21 February 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1439

Resolution of the Navier–Stokes equations on block-structured meshes

C. Romé, S. Glockner*, † and J. P. Caltagirone

Laboratoire TREFLE, CNRS-UMR 8508, 16 av. Pey-Berland, 33607 PESSAC Cedex, France

SUMMARY

We propose a non-iterative method to connect non-matching block-structured meshes applied to the resolution of the Navier–Stokes equations. The present article gives a complete description of the method based on the implicit treatment of the connecting condition. We also extend it to curvilinear meshes. The spatial convergence order of the method is shown to be two and the approach is validated on different 2D laminar flows frequently found in the literature. Copyright © 2007 John Wiley & Sons, Ltd.

Received 12 September 2006; Revised 1 December 2006; Accepted 6 December 2006

KEY WORDS: block-structured meshes; Navier–Stokes; non-matching; non-conforming; overlapping; interpolation

1. INTRODUCTION

A key step in a numerical simulation is the generation of a mesh, the first function of which is to accurately follow the contour lines of the geometry. Generally speaking, meshes can be distinguished as structured or unstructured, Cartesian or curvilinear, orthogonal or not, monoblock or multiblock, etc. The starting point of this work is a computational fluid dynamics (CFD) code written for one curvilinear, structured and orthogonal block. To extend its use to complex geometries, it was natural to work on block-structured meshes. The freedom given by non-matching interfaces allows geometry curves to be followed more precisely thanks to the fact that the blocks are independent.

We first present the numerical context of this work by describing the models and numerical methods of our CFD code Aquilon (Aq. on the figures and tables) and continue with a presentation of the domain decomposition framework.

Next, we propose a method, first introduced in [1], of connecting non-conforming meshes. It is non-iterative and relies on the implicit treatment of the interface condition. We apply it to

*Correspondence to: S. Glockner, Laboratoire TREFLE, CNRS-UMR 8508, 16 av. Pey-Berland, 33607 PESSAC Cedex, France.
†E-mail: glockner@enscpb.fr

the Navier–Stokes equations for laminar divergence-free flows in the framework of the augmented Lagrangian method. Special attention is paid to the interpolation techniques used. For a better comprehension of the implementation of the method, we present it first for Cartesian block-structured meshes and then generalize to orthogonal curvilinear coordinate systems.

Finally, we show that the spatial convergence order of the method is two and validate it on academic test cases such as the flow over a backward facing step, the lid-driven cavity flow and the flow around a cylinder (stationary or not).

2. NUMERICAL CONTEXT

2.1. Numerical methods

We are interested in the incompressible Navier–Stokes equations:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u})\right) = -\nabla p + \nabla \cdot \mu(\nabla \mathbf{u} + \nabla^{\mathrm{t}}\mathbf{u}) \tag{2}$$

The implementation of the boundary conditions is done thanks to a penalization technique which derives from a Fourier boundary condition found in thermal science [2] and expressed by

$$\frac{\partial T}{\partial n} + Bi(T - T_\infty) = 0 \tag{3}$$

where T is the temperature, T_∞ a reference temperature, and Bi the Biot number. According to the values of Bi, one can impose a Dirichlet condition ($Bi = \infty$), a Neumann condition ($Bi = 0$), or a Fourier condition if the value of Bi is bounded between 0 and ∞.

A penalization term is added to the Navier–Stokes equations, written $\mathbf{BIU}(\mathbf{u} - \mathbf{u}_\infty)$ where $\mathbf{BIU}$ is an order 2 diagonal tensor of component (BIU_u, BIU_v), with $0 \leqslant BIU_u \leqslant \infty$ and $0 \leqslant BIU_v \leqslant \infty$.

Thus, the Navier–Stokes system can be written as

$$\begin{gathered} \nabla \cdot \mathbf{u} = 0 \quad \text{on } \Omega \\ \rho\left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u})\right) + \mathbf{BIU}(\mathbf{u} - \mathbf{u}_\infty) = -\nabla p + \nabla \cdot \mu(\nabla \mathbf{u} + \nabla^{\mathrm{t}}\mathbf{u}) \quad \text{on } \Omega \\ (\nabla \mathbf{u}) \cdot \mathbf{n} = 0 \quad \text{on } \partial\Omega \end{gathered} \tag{4}$$

Time implicit discretization is used and the inertial term is linearized at order one. The resolution of the Navier–Stokes equations requires a consistent pressure and a divergence-free velocity field to be obtained at each time step. This coupling is difficult to treat for incompressible flows where pressure does not appear explicitly in the mass conservation equations. Two classes of methods exist, depending on the fact whether the Navier–Stokes operator is split or not:

- None time splitting and exact space methods:
  - Coupled resolution of velocity and pressure. It is tedious and complex because the matrices are very ill conditioned.
  - Artificial compressibility method described by Peyret and Taylor [3]. It takes into account a perturbation parameter which acts on pressure in the mass continuity equation. Pressure

DOI: 10.1002/fld

is eliminated from the momentum equations. This technique often gives ill-conditioned matrices and needs an iterative method such as the Uzawa algorithm [4].
  - Implicit approaches which are minimization methods under constraint. The augmented Lagrangian method is one of them and has been analysed by Fortin and Glowinski [5]. It is used in our code for laminar, incompressible, turbulent, multiphase flows [2, 6–9] and is described later on.

- Time splitting methods. These consist of a prediction/diffusion step and a correction/projection one for the velocity and pressure field. The following methods are often distinguished:
  - Projection methods introduced by Chorin [10, 11] and their variants [12, 13].
  - Incremental methods for the pressure correction introduced by Goda [14] and then improved by Timmermans *et al.* [15].
  - Prediction/correction algorithms such as SIMPLE (semi implicit method for pressure linked equations) or SIMPLER (SIMPLE revised) [16, 17] which use a pressure correction equation to modify the velocity field.

2.1.1. The augmented Lagrangian method. The book of Fortin and Glowinsky [5] presents numerous applications of this method applied to the resolution of partial differential equations, especially the Stokes and Navier–Stokes problems. The Stokes system is formulated as a velocity–pressure minimization–maximization problem requiring the computation of a saddle point ($\mathbf{u}$, p) associated with the augmented Lagrangian of the problem. The pressure is considered as a Lagrange multiplier and the incompressibility constraint is introduced implicitly into the momentum equations. Then, the saddle-point ($\mathbf{u}$, p) is computed using an iterative Uzawa algorithm. On a more formal point of view, if

$$L^2(\Omega) = \left\{ f, \int_\Omega |f|^2 \, \mathrm{d}\Omega < \infty \right\}$$

$$H_0^1(\Omega) = \left\{ f \in L^2(\Omega), \frac{\partial f}{\partial x_i} \in L^2(\Omega), \ i \in [1, 3], \ f|_{\partial\Omega} = 0 \right\}$$

a functional $J(\mathbf{v})$ is defined for each $\mathbf{v} \in H_0^1(\Omega)$, resulting from the weak formulation of the momentum equations, and has to be minimized under the constraint

$$M = \{\mathbf{v} \in H_0^1(\Omega), \ \nabla \cdot \mathbf{v} = 0\}$$

which is equivalent to find $\mathbf{u} \in M$ such as

$$J(\mathbf{u}) = \min_{\mathbf{v} \in M}(J(\mathbf{v})) \tag{5}$$

The constraint is then satisfied thanks to a Lagrange multiplier q, transforming the problem with the constraint M to a problem without constraint. The Lagrangian is defined as

$$\begin{aligned} &L : M \times L^2 \to \Re \\ &(\mathbf{v}, q) \mapsto L_r(\mathbf{v}, q) = J(\mathbf{v}) - \int_\Omega q \nabla \cdot \mathbf{v} \, \mathrm{d}\Omega \end{aligned} \tag{6}$$

DOI: 10.1002/fld

The minimization problem consists of finding the saddle-point $(\mathbf{u}, p) \in M \times L^2$ of the Lagrangian L which verifies

$$\begin{aligned} L(\mathbf{u}, p) &= \min_{\mathbf{v}\in M}\left(\max_{q\in L^2}(L(\mathbf{v}, q))\right) \\ &= \max_{q\in L^2}\left(\min_{\mathbf{v}\in M}(L(\mathbf{v}, q))\right) \end{aligned} \tag{7}$$

To improve the convergence properties, for each $(\mathbf{v}, q) \in M \times L^2$, the augmented Lagrangian is defined as

$$\begin{aligned} &L_r : M \times L^2 \to \Re \\ &(\mathbf{v}, q) \mapsto L_r(\mathbf{v}, q) = J(\mathbf{v}) - \int_\Omega q\nabla\cdot\mathbf{v}\,\mathrm{d}\Omega + \int_\Omega \frac{\mathrm{d}r}{2}|\nabla\cdot\mathbf{v}|^2\,\mathrm{d}\Omega \end{aligned} \tag{8}$$

A saddle-point of L is also a saddle-point of L_r (and reciprocally). It is demonstrated for the Stokes problem and admitted for the Navier–Stokes equations, that, coming back to the strong formulation, the saddle-point is also the solution of the system

$$\begin{aligned} &\nabla\cdot\mathbf{u} = 0 \\ &\rho\left(\frac{\partial\mathbf{u}}{\partial t} + \nabla\cdot(\mathbf{u}\otimes\mathbf{u})\right) + \mathbf{BIU}(\mathbf{u}-\mathbf{u}_\infty) = -\nabla p + \nabla\cdot\mu(\nabla\mathbf{u} + \nabla^{\mathrm{t}}\mathbf{u}) - \mathrm{d}r\nabla(\nabla\cdot\mathbf{u}) \end{aligned} \tag{9}$$

The calculation of the solution is done thanks to the Uzawa iterative method [5] which generates the following k-iterations $(\mathbf{u}^k, p^k)$ that can be stopped by setting a criterion based on the divergence of the velocity field, i.e. $\|\nabla\cdot\mathbf{u}\| \leqslant \varepsilon$, with ε small

Initialization
$\mathbf{u}^{k=0} = \mathbf{u}^n$ and $p^{k=0} = p^n$
Iterations
for $k = 0, K-1$
computation of $\mathbf{u}^{k+1}$ *solution of*:

$$\begin{aligned} &\rho^n\left(\frac{\mathbf{u}^{k+1}}{\Delta t} + \nabla\cdot(\mathbf{u}^{k+1}\otimes\mathbf{u}^k) - \mathbf{u}^{k+1}\nabla\cdot\mathbf{u}^k\right) + \mathbf{BIU}(\mathbf{u}^{k+1}-\mathbf{u}_\infty) \\ &\quad = -\nabla p^k + \nabla\cdot\mu(\nabla\mathbf{u}^{k+1} + \nabla^t\mathbf{u}^{k+1}) + dr\nabla(\nabla\cdot\mathbf{u}^{k+1}) + \rho\frac{\mathbf{u}^k}{\Delta t} \end{aligned} \tag{10}$$

updating of p^{k+1} *with*:
$p^{k+1} = p^k - dp\nabla\cdot\mathbf{u}^{k+1}$
Solution
$\mathbf{u}^{n+1} = \mathbf{u}^K$, $p^{n+1} = p^K$

It has been demonstrated [5] for the Stokes equations that the Uzawa algorithm converges under the condition $0 < \mathrm{d}p \leqslant 2\mathrm{d}r$. The use of high values of $\mathrm{d}r$ (up to 10^4, 10^5) increases the convergence

DOI: 10.1002/fld

of the algorithm but leads to ill-conditioned matrix associated to the discretization of the problem. Nevertheless, associated with a direct solver, this can be a judicious choice.

From a practical point of view, for the test cases described below, we have chosen $\mathrm{d}p = \mathrm{d}r$ and one to four iterations of the Uzawa algorithm. For a stationary problem, one iteration is enough, convergence being done on the time iterations. For unstationary problems, we have used four iterations at each time step to reach divergence level below 10^{-5}.

Contrary to the time splitting methods, this algorithm does not even suffer from error introduced by the splitting of the operators. The precision of the method is only driven by the time and space schemes precisions. Finally, it has to be noted that no boundary condition is needed for the pressure, which is calculated explicitly in the whole domain.

Verifying the incompressible constraint at the computer precision can be expensive with regard to the CPU time. When the Uzawa algorithm is applied with few iterations, the resulting divergence of the velocity field is small but not identically null. To circumvent this drawback, we have used the vectorial projection method introduced by Caltagirone and Breil [18].

2.1.2. Vectorial projection method. Velocity $\mathbf{u}^{n+1}$ is decomposed into $\mathbf{u}^* + \mathbf{u}'$ where $\mathbf{u}^*$ is computed by the augmented Lagrangian step and is considered as a predictive divergence-free velocity. The latter verifies

$$\rho^n \left(\frac{\mathbf{u}^{n+1}}{\Delta t} + \nabla \cdot (\mathbf{u}^{n+1} \otimes \mathbf{u}^n) - \mathbf{u}^{n+1} \nabla \cdot \mathbf{u}^n \right) + \mathbf{BIU}(\mathbf{u}^{n+1} - \mathbf{u}_\infty)$$

$$= -\nabla p^n + \nabla \cdot \mu(\nabla \mathbf{u}^{n+1} + \nabla^{\mathrm{t}} \mathbf{u}^{n+1}) + \mathrm{d}r \nabla(\nabla \cdot \mathbf{u}^{n+1}) + \rho \frac{\mathbf{u}^n}{\Delta t} \tag{11}$$

By replacing $\mathbf{u}^{n+1}$ by $\mathbf{u}^* + \mathbf{u}'$ in (11), and with $\mathrm{d}r \to \infty$, we obtain

$$\nabla(\nabla \cdot \mathbf{u}^{n+1}) = \nabla(\nabla \cdot \mathbf{u}^*) + \nabla(\nabla \cdot \mathbf{u}') = 0 \tag{12}$$

or

$$\nabla(\nabla \cdot \mathbf{u}') = -\nabla(\nabla \cdot \mathbf{u}^*) \tag{13}$$

Both velocities $\mathbf{u}^{n+1}$ and $\mathbf{u}^*$ satisfy the physical boundary conditions. We deduce that the boundary conditions of $\mathbf{u}'$ are homogeneous.

The existence of the solution of the algebraic square system associated with Equation (13) is ensured by the fact that the second member is in the range of the discrete operator. In addition, one can check that the image is orthogonal to the kernel and thus their intersection is reduced to the null vector. The latter property makes possible to ensure the uniqueness of the solution, at least when the system is solved by an iterative method of Krylov type (BiCGStab in our case) while starting the iterative algorithm by an initial guess in the range of the operator (zero for example).

2.1.3. Spatial discretization. This is based on the finite volume method on velocity–pressure staggered grids of the Marker and Cells type [19]. Pressure unknowns are associated to the cell vertices, whereas velocity component are face centred. The centred scheme of order 2 is used in this work for the inertial and constraint terms.

A Dirichlet boundary condition on the normal component of the velocity field has to be verified on the pressure nodes which belong to the physical boundary. The existence of velocity points outside

DOI: 10.1002/fld

the domain requires to write the penalization term of the Navier–Stokes equations $\mathbf{BIU}(\mathbf{u}-\mathbf{u}_\infty)$ as $\mathbf{BIU}(\mathbf{f}_{\text{lim}}(\mathbf{u})-\mathbf{u}_\infty)$ where $\mathbf{f}_{\text{lim}}$ is a linear combination of the discrete unknowns (a centred scheme is used).

The linear system is sparse, composed of nine diagonals in 2D. The multifrontal spare direct solver MUMPS [20] is used to solve the Navier–Stokes linear system and a BiCGStab iterative solver, ILU preconditioned, is used for the linear system associated with the vectorial projection method.

2.2. *On some domain decomposition methods*

In this numerical context, it is natural to be interested in domain decomposition techniques which offer a general framework for the treatment of block interfaces. Domain decomposition is based on the idea that the solution of a global problem can be obtained by assembling solutions of smaller, more regular ones. There are two fields of application: the first concerns the resolution of large linear systems, the second multiphysical problems (scale change, fluid/structure coupling, heterogeneous domains, etc.).

The principle of domain decomposition is quite simple. The domain is divided into sub-domains on which the original problem is written again in such a way that solutions are coupled thanks to appropriate conditions on the interfaces. Iterative techniques are usually used to solve the problem. The main example is the Dirichlet/Neumann algorithm for the Laplacian problem where the transmission condition connects the variables and their normal derivatives [21].

Generally speaking, domain decomposition deals with overlapping (or not) and conforming (or not) sub-domains. Historically, H. A. Schwarz was the originator of these techniques, with the additive and multiplicative algorithms. The main drawback of the Schwarz method is that overlapping is necessary for convergence. An improvement consists in replacing overlapping by a supplementary condition, the literature for which is very rich. The Dirichlet condition can be replaced by a Robin condition [22].

Our strategy is to work with overlapping. The challenge is then to find an adequate projection operator on the interfaces between the blocks. In this field, the mortar method has been proposed [23–25]. A method, based on the Robin condition, also deals with these problems [26, 27].

Another approach, the Chimera method [28, 29], is generally applied to aeronautical problems. Originally, this method aims to simplify mesh generation, coupling between blocks being achieved by an iterative Dirichlet/Dirichlet condition. Other interface conditions have also been proposed [30] (Dirichlet/Neumann, Dirichlet/Robin). Brezzi *et al*. [31] have proved that the Chimera method is a variant of the Schwarz algorithm.

Non-matching meshes raise the classical question of interpolation. This difficulty becomes important when it is necessary to interpolate under constraint (here $\nabla\cdot\mathbf{u}=0$). The question is to know if precise and conservative operators exist. Generally, interpolation is conservative when it is based on finite volume techniques [32]. Fluxes (of mass, momentum or energy) through interfaces are calculated using local balance with a neighbouring block or a projection. Non-conservative interpolation is based on mathematical interpolation of the variables (Lagrange interpolation for instance). Some authors who use a non-conservative interpolation have shown that mass conservation is directly linked to the order of the interpolation method [33]. Although the conservative/non-conservative dilemma remains important, it has been minimized by Meakin [34]. According to him, what is important in the treatment of the interface is the sharpness of the grid. A precise non-conservative method can give better results than a less precise conservative one.

DOI: 10.1002/fld

2.2.1. Towards an implicit connection between blocks. We have chosen an implicit resolution of the conservation equations and the connecting conditions at the interface. The method is built on non-conservative interpolation of the variables. Polynomial coefficients are present in the global linear system and couple, at the same time, the solution both at each block and at the interfaces. This method can be seen as a non-iterative, implicit Dirichlet/Dirichlet condition for which minimal overlapping is necessary.

3. AN IMPLICIT METHOD FOR CONNECTING BLOCKS

Block-structured meshes allow a domain to be cut out according to the geometric contour. Generally, blocks have different orientations from one another, which induces a discontinuity of the mesh lines through the interfaces. We must connect the blocks by transferring the missing information from block to block. Polynomial interpolations are built and integrated as special boundary conditions in the global linear system. The proposed solution for transferring the information between blocks consists in implicitly interpolating a field ϕ at the nodes situated on the interfaces by using the nodes of the adjacent block. This interpolation can be considered as a new implicit boundary condition used for the discretization of the equation at the nodes strictly inside the different blocks (see Figure 1).

We will first present the method in the case of Cartesian blocks with parallel interfaces, then extend it to the case of Cartesian blocks of any orientation, and finally generalize it to curvilinear grids.

3.1. Parallel blocks

The variable ϕ (u or v) defined on block (b) is interpolated, which gives the new boundary condition on block (a). We recall that the penalization term of the momentum equations used for the boundary conditions is written $\mathbf{BIU}(\mathbf{f}_{\text{lim}}(\mathbf{u}) - \mathbf{u}_\infty)$. Here, the expression of $\mathbf{u}_\infty$ at the nodes of an interface is the result of the interpolation of the values of $\mathbf{u}$ on the adjacent block. The interpolation of the normal component of the velocity field to the interface is performed on pressure nodes, whereas velocity nodes are used for interpolating the tangential component (see Figure 2).

Interpolation is based on the construction of a polynomial basis of a given order. For instance, if ϕ represents u or v, the interpolation of ϕ at a point $M_0(x_0, y_0)$ that belongs to a block (a) (or

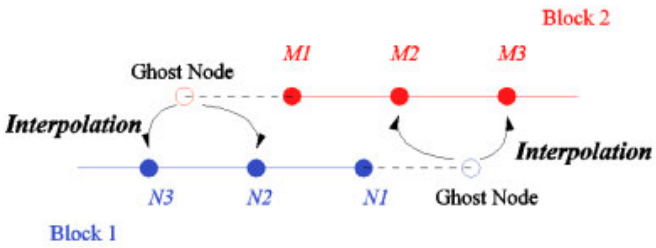


Figure 1. Connecting method for non-conforming meshes.

DOI: 10.1002/fld

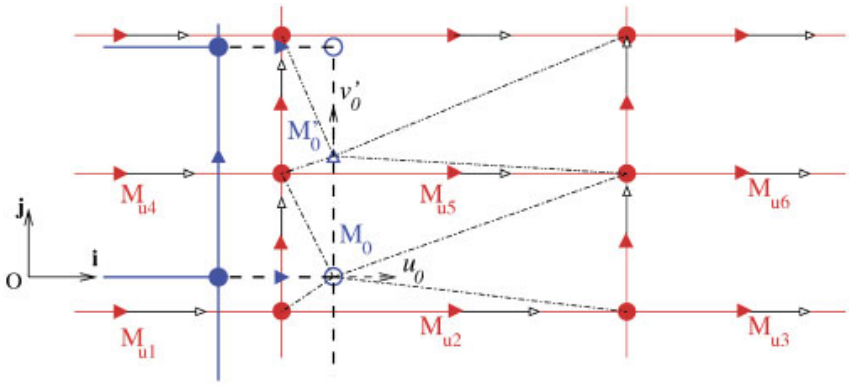


Figure 2. Interpolation of the velocity vector on a staggered grid.

at a point $M'_0(x_0, y_0)$ for v), is obtained from the values of ϕ on block (b) by the relation:

$$\phi^{(a)}(x_0, y_0) = f_{\text{int}}(\phi_i^{(b)}) = \sum_{i=1}^{N} F_i(x_0, y_0)\phi_i^{(b)}(x_i, y_i) \tag{14}$$

in which the F_i are the values at the point (x_0, y_0) of the N polynomials that constitute the polynomial basis, and $\phi_i^{(b)}$ are the N values of the field ϕ at the interpolation nodes $M_i(x_i, y_i)$ of block (b).

The interpolations must now be constructed locally to each node of the interface. The technique consists in building a canonical basis of Q-type polynomials of order d thanks to the neighbouring nodes of $M_0(x_0, y_0)$. The number of nodes required depends on the order of the chosen polynomial.

In order to reduce the values of the coefficient of the polynomial, M_0 is chosen as the centre of the frame. A polynomial $Q_i^{(d)}$ built using the M_i nodes, $1 \leqslant i \leqslant (d+1)^2$ is written

$$Q_i^{(d)}(x - x_0, y - y_0) = \sum_{m=0}^{d} \sum_{n=0}^{d} a_{mni}(x - x_0)^m (y - y_0)^n \tag{15}$$

It has the following properties:

$$\forall i, j, \quad 1 \leqslant i, \quad j \leqslant (d+1)^2, \quad Q_i^{(d)}(x - x_0, y - y_0) = \delta_{ij} \tag{16}$$

A $(d+1)^2 x (d+1)^2$ linear system is built (17), Equation (16) being a line of the matrix.

$$\begin{bmatrix} a_{001} & \cdot & a_{mn1} & \cdot & a_{dd1} \\ \cdot & & \cdot & & \cdot \\ a_{00i} & & a_{mni} & & a_{ddi} \\ \cdot & & \cdot & & \cdot \\ a_{00(d+1)} & \cdot & a_{mn(d+1)} & \cdot & a_{dd(d+1)} \end{bmatrix}, \quad B = \begin{bmatrix} 1 & \cdot & 0 & \cdot & 0 \\ \cdot & & \cdot & & \cdot \\ 0 & \cdot & 1 & \cdot & 0 \\ \cdot & & \cdot & & \cdot \\ 0 & \cdot & 0 & \cdot & 1 \end{bmatrix} \tag{17}$$

DOI: 10.1002/fld

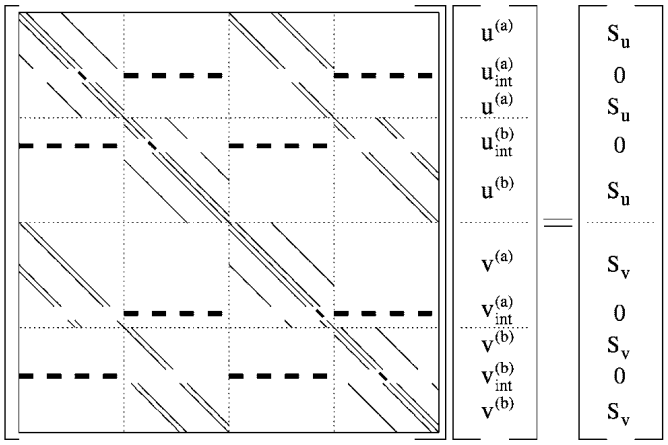


Figure 3. Representation of the Navier–Stokes matrix on 2 blocks.

with B given by

$$\begin{bmatrix} (x_1-x_0)^0(y_1-y_0)^0 & \cdot & (x_i-x_0)^0(y_i-y_1)^0 & \cdot & (x_{d+1}-x_0)^0(y_{d+1}-y_0)^0 \\ \cdot & & \cdot & & \cdot \\ (x_1-x_0)^m(y_1-y_0)^n & & (x_i-x_0)^m(y_i-y_1)^n & & (x_{d+1}-x_0)^0(y_{d+1}-y_0)^n \\ \cdot & & \cdot & & \cdot \\ (x_1-x_0)^d(y_1-y_0)^d & \cdot & (x_i-x_0)^d(y_i-y_1)^d & \cdot & (x_{d+1}-x_0)^0(y_{d+1}-y_0)^d \end{bmatrix} \tag{18}$$

The inversion of the linear system (one for each interface node) is carried out once during the preparation step of a simulation (before the time loop resolution). The value of the field ϕ at node $M_0(x_0, y_0)$ is then given by

$$\phi^{(a)}(x_0, y_0) = \sum_{i=1}^{(d+1)^2} Q_i^{(d)}(0,0)\phi^{(b)}(x_i, y_i) \tag{19}$$

It can be seen that the polynomial coefficients $Q_i^d(0,0)$ are not time dependent. By writing Equation (19) at time $n+1$, the $Q_i^{(d)}$ can be placed in the linear system (see Figure 3) of the momentum equations. Thus, on a matrix line corresponding to an interface node, non-zero elements are the main diagonal and the elements with a column number correspond to the unknowns used to interpolate the field.

3.2. Cartesian blocks with any orientation

In the general case, the interface is not parallel to the mesh lines of the adjacent block. We consider now the case of two blocks (a) and (b) with non-collinear local frames $(i^{(a)}, j^{(a)})$ and $(i^{(b)}, j^{(b)})$ (see Figure 4). Both components of the velocity vector are thus necessary for the interpolation of each component. Frame changes are then required. As the velocity vector can be defined at

DOI: 10.1002/fld

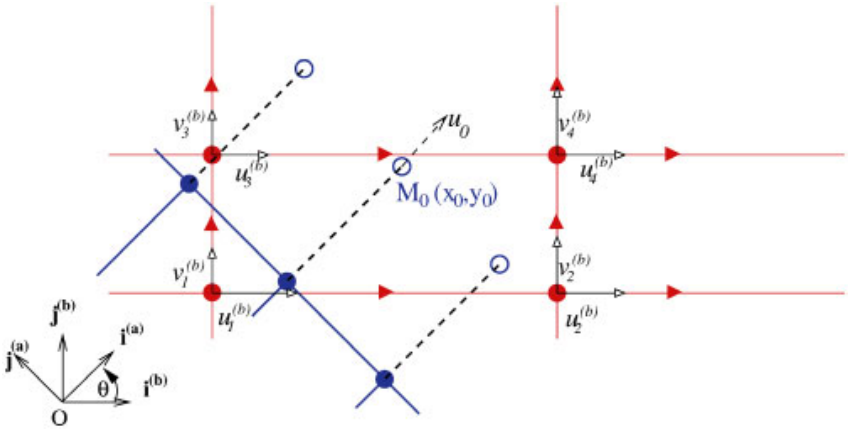


Figure 4. Interpolation of the velocity vector with frame change.

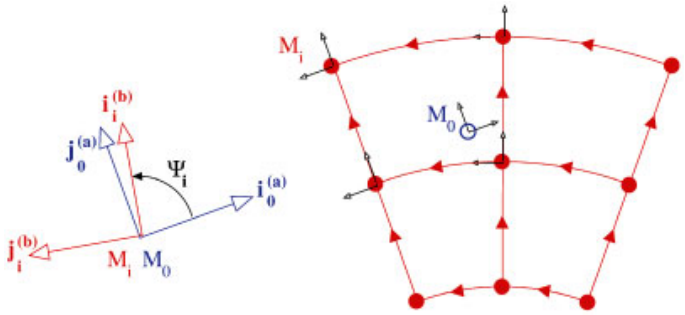


Figure 5. Interpolation of the velocity vector (curvilinear mesh).

the pressure node, the interpolation is performed at these nodes. Thus, for each component, the number of interpolations is doubled compared to the previous case:

$$\begin{aligned} u^{(a)}(x_0, y_0) &= \cos\theta \sum_{i=1}^{(d+1)^2} Q_i^{(d)}(0,0) u_i^{(b)}(x_i, y_i) - \sin\theta \sum_{i=1}^{(d+1)^2} Q_i^{(d)}(0,0) u_i^{(b)}(x_i, y_i) \\ v^{(a)}(x_0', y_0') &= \sin\theta \sum_{i'=1}^{(d+1)^2} Q_{i'}^{(d)}(0,0) v_{i'}^{(b)}(x_{i'}, y_{i'}) + \cos\theta \sum_{i'=1}^{(d+1)^2} Q_{i'}^{(d)}(0,0) v_{i'}^{(b)}(x_{i'}, y_{i'}) \end{aligned} \tag{20}$$

with θ the angle between the frames $(i^{(a)}, j^{(a)})$ and $(i^{(b)}, j^{(b)})$, and $u^{(b)}$, $v^{(b)}$ the velocity components at the pressure node.

3.3. Curvilinear blocks

In the case of curvilinear blocks, the velocity field is defined on a local frame that is different at each point of the domain (see Figure 5).

DOI: 10.1002/fld

It is necessary to make frame changes locally to each node. Let $(i^{(a)}, j^{(a)})$ be the frame of an interface node. We have

$$\begin{aligned} u^{(a)}(x_0, y_0) &= \sum_{i=1}^{(d+1)^2} Q_i^{(d)}(0,0) u_0^{(b)}(x_i, y_i) \\ v^{(a)}(x_0', y_0') &= \sum_{i'=1}^{(d+1)^2} Q_{i'}^{(d)}(0,0) v_0^{(b)}(x_{i'}, y_{i'}) \end{aligned} \tag{21}$$

where $u_0^{(b)}$ and $v_0^{(b)}$ are expressed in the frame $(i^{(a)}, j^{(a)})$. The frame change from $(i^{(b)}, j^{(b)})$ to $(i^{(a)}, j^{(a)})$ gives

$$\begin{aligned} u_0^{(b)}(x_i, y_i) &= \cos(\psi_i) u_i^{(b)}(x_i, y_i) - \sin(\psi_i) v_i^{(b)}(x_i, y_i) \\ v_0^{(b)}(x_i', y_i') &= \sin(\psi_i) v_{i'}^{(b)}(x_{i'}, y_{i'}) + \cos(\psi_i) v_{i'}^{(b)}(x_{i'}, y_{i'}) \end{aligned} \tag{22}$$

We obtain

$$\begin{aligned} u^{(a)}(x_0, y_0) &= \sum_{i=1}^{(d+1)^2} Q_i^{(d)}(0,0) \cos(\psi_i) u_i^{(b)}(x_i, y_i) - \sin(\psi_i) v_i^{(b)}(x_i, y_i) \\ v^{(a)}(x_0', y_0') &= \sum_{i'=1}^{(d+1)^2} Q_{i'}^{(d)}(0,0) \sin(\psi_i) v_{i'}^{(b)}(x_{i'}, y_{i'}) + \cos(\psi_i) v_{i'}^{(b)}(x_{i'}, y_{i'}) \end{aligned} \tag{23}$$

Finally, the expression for the penalization term on an interface node is given by $\mathbf{BIU}(\mathbf{f}_{\text{lim}}(\mathbf{u}) - \mathbf{f}_{\text{int}}(\mathbf{u}))$, with

$$\mathbf{f}_{\text{int}}(\mathbf{u}) = \begin{pmatrix} \sum_{i=1}^{N} \alpha_{iu} u^{(2)}(x_i, y_i) + \sum_{j=1}^{N} \beta_{iu} v^{(2)}(x_i, y_i) \\ \sum_{i=1}^{N} \alpha_{iv} u^{(2)}(x_i, y_i) + \sum_{j=1}^{N} \beta_{iv} v^{(2)}(x_i, y_i) \end{pmatrix} \tag{24}$$

Coefficients α_{iu}, α_{iv}, β_{iu} and β_{iv} contain the expressions of frame changes, the values of interpolation polynomials, and the interpolation coefficients of the velocity on the pressure grid.

Pressure is also interpolated to ensure continuity of the pressure (to the order of the interpolation polynomial).

3.4. Mass conservation

The non-conservative interpolation produces non-conservation of the mass between the blocks. Divergence is not null, especially at the interface nodes. We observe two behaviours: in the case of confined flows (driven cavity problem for instance), divergence levels are almost constant on each block; in the case of open flows (flow around a backward facing step, around a cylinder for instance), divergence is null on the whole domain except at the interface nodes. Even if, as we will see, the results of a series of test cases are good, this remains a drawback of the method. We propose to apply a correction to the method locally to each block, the mass non-conservation on the whole domain remaining unchanged.

DOI: 10.1002/fld

From the predictive velocity field $\mathbf{u}^*$ given by the Uzawa algorithm, we modify the boundary conditions associated with the vectorial projection method on each block using the Green–Ostrogradski formula (25) which links the integral of the velocity divergence over the whole block to the flow rate at the interface

$$\iiint_{\Omega_i} \nabla \cdot \mathbf{u}^* \, \mathrm{d}v = \iint_{\partial\Omega_i} \mathbf{u}^* \cdot \mathbf{n} \, \mathrm{d}s \tag{25}$$

The idea is to recover flow rate conservation on the boundary of each block by applying a velocity correction $\tilde{\mathbf{u}}^*$ such that

$$\iint_{\partial\Omega_i} (\mathbf{u}^* + \tilde{\mathbf{u}}^*) \cdot \mathbf{n} \, \mathrm{d}s = 0 \tag{26}$$

With Equation (25), we have

$$\iint_{\partial\Omega_i} \tilde{\mathbf{u}}^* \cdot \mathbf{n} \, \mathrm{d}s = - \iiint_{\Omega_i} \nabla \cdot \mathbf{u}^* \, \mathrm{d}v \tag{27}$$

The correction is applied only at the interfaces Γ_i of each block because boundary conditions are already satisfied at the physical boundaries. Therefore, locally to each block, we have

$$\iint_{\Gamma_i} \tilde{\mathbf{u}}^* \cdot \mathbf{n} \, \mathrm{d}s = - \iiint_{\Omega_i} \nabla \cdot \mathbf{u}^* \, \mathrm{d}v = -D_{\Omega_i} \tag{28}$$

Thus, we can consider a homogeneous or a local correction at each interface node.

If $\tilde{\mathbf{u}}_l^*$ is the correction applied to the velocity $\mathbf{u}_l^*$ at an interface node, $\mathbf{n}_l$ the outward normal and S_l the local section, the homogeneous correction is written

$$\tilde{\mathbf{u}}_l^* \cdot \mathbf{n}_l = \frac{D_{\Omega_i}}{\sum_{j=1}^{N} S_j} \tag{29}$$

And the local correction depending on the local flow rate is given by

$$\tilde{\mathbf{u}}_l^* \cdot \mathbf{n}_l = \frac{D_{\Omega_i} (\mathbf{u}_l^* \cdot \mathbf{n}_l S_l)}{S_j \sum_{j=1}^{N} \mathbf{u}_l^* \cdot \mathbf{n}_l S_j} \tag{30}$$

This correction can be applied to the vectorial projection method to ensure $\nabla \cdot (\mathbf{u}^* + \mathbf{u}') = 0$ locally to each block. The correction $\tilde{\mathbf{u}}^*$ is used to set a Dirichlet boundary condition on the interface (see Equation (31)). A few iterations of a BiCGStab with ILU preconditioning are enough to reach null divergence on each block.

Solution of the augmented Lagrangian $\mathbf{u}^* = \mathbf{u}^{\mathbf{K}}$ and $\mathbf{p}^* = \mathbf{p}^{\mathbf{K}}$
Vectorial Projection

- at each interface Γ_i of section S_i associated with a block Ω_i, calculation of $\tilde{\mathbf{u}}'_{\infty i}$ from equation:

$$\tilde{\mathbf{u}}'_{\infty i} = - \frac{D_{\Omega_i}}{S_i} \mathbf{n}_i \tag{31}$$

- calculation of $\mathbf{u}'$ solution of:

$$\nabla(\nabla \cdot \mathbf{u}') + \mathbf{BIU}(\mathbf{f}_{\lim}(\mathbf{u}') - \tilde{\mathbf{u}}'_{\infty}) = - \nabla(\nabla \cdot \mathbf{u}^*)$$

DOI: 10.1002/fld

4. CONVERGENCE STUDY

4.1. Poiseuille flow (Cartesian meshes)

Simulations were carried out on the various block-structured meshes shown in Figure 6. Results are given for convergence criteria below 10^{-10} for stationarity.

With a Q^1 polynomial, the solution of Poiseuille flow, which is of order 2, cannot be reproduced without errors (see Table I). With Q^2 or Q^3 polynomials, errors are close to the computer accuracy.

4.2. Green–Taylor vortex

The main interest of this flow is that, unlike in Poiseuille flow, the inertial term is not null. The Green–Taylor vortex is modified to obtain a stationary solution not identically null. The momentum equations are enriched by the following source term:

$$\mathbf{S} = \begin{cases} -\dfrac{\pi^2\mu}{2H^2}\cos\left(\dfrac{\pi x}{2H}\right)\sin\left(\dfrac{\pi y}{2H}\right) \\ -\dfrac{\pi^2\mu}{2H^2}\sin\left(\dfrac{\pi x}{2H}\right)\cos\left(\dfrac{\pi y}{2H}\right) \end{cases} \tag{32}$$

The solution is

$$\begin{aligned} u(x,y,t) &= -\cos\left(\frac{\pi x}{2H}\right)\sin\left(\frac{\pi y}{2H}\right)(1-\mathrm{e}^{-\pi^2\upsilon t/2H^2}) \\ v(x,y,t) &= -\sin\left(\frac{\pi x}{2H}\right)\cos\left(\frac{\pi y}{2H}\right)(1-\mathrm{e}^{-\pi^2\upsilon t/2H^2}) \\ p(x,y,t) &= -\frac{\rho}{2}\left(\cos^2\left(\frac{\pi x}{2H}\right)+\cos^2\left(\frac{\pi y}{2H}\right)\right)(1-2\mathrm{e}^{-\pi^2\upsilon t/2H^2}+\mathrm{e}^{-\pi^2\upsilon t/H^2}) \end{aligned} \tag{33}$$

The boundary conditions are obtained directly from the analytical solution and are modified at each time step. The test case was run with two non-conforming structured blocks (see Figure 7).

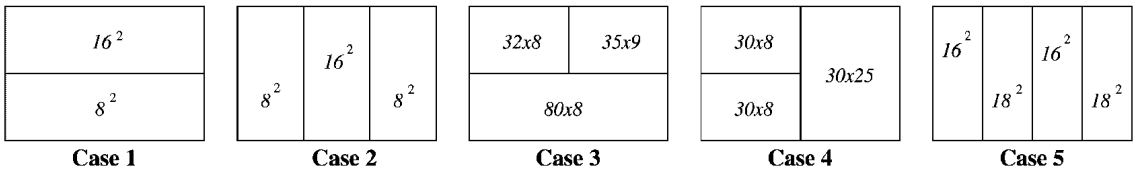


Figure 6. Representation of the different blocks for Poiseuille flow study; the number of elements per block is indicated.

Table I. L_2 norm of the error for Poiseuille flow.

	Case 1	Case 2	Case 3	Case 4	Case 5
Q^1	1.98×10^{-3}	7.81×10^{-3}	3.63×10^{-2}	2.05×10^{-2}	1.04×10^{-3}
Q^2	1.02×10^{-10}	4.36×10^{-13}	3.18×10^{-10}	2.32×10^{-12}	4.15×10^{-13}
Q^3	1.75×10^{-10}	2.71×10^{-13}	7.73×10^{-9}	30.3×10^{-12}	5.66×10^{-13}

DOI: 10.1002/fld

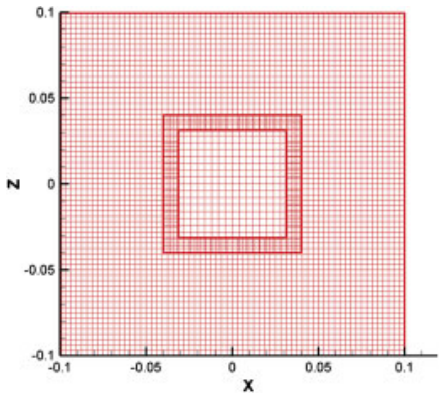


Figure 7. Block-structured mesh example for the Green–Taylor flow study.

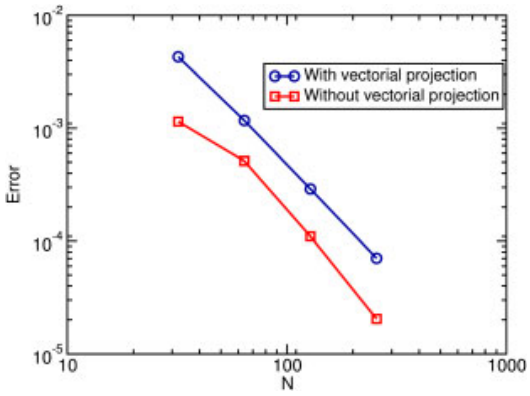


Figure 8. L_2 norm of the error as a function of mesh size for the Green–Taylor flow study.

Table II. L_∞ norm of the error on velocity field at $T = 1$ s.

Δt (s)	0.1	0.05	0.01	0.005	0.001
L_∞ error	7.72×10^{-4}	3.87×10^{-4}	7.93×10^{-5}	4.08×10^{-5}	1.00×10^{-5}
Local slope (in log/log scale)		0.99	0.98	0.96	0.87

The domain is a square of side length 0.2 m ($H = 0.1$). The fluid used has a density of $1.176\,\mathrm{kg/m^3}$ and a viscosity of 1.85×10^{-5} Pa s.

Simulations were performed using or not using the vectorial projection method. We first focus on the stationary solution. With Q^2 interpolation polynomials, the spatial convergence order is two, as shown in Figure 8. Validation on pressure was not established. Indeed, divergence at the end of the Uzawa algorithm was not null (around 10^{-6}) and accumulated in the pressure.

Then, a time convergence study has been carried out (a two blocks non-conforming mesh size of 4300 nodes is used). Errors on velocity field at time $T = 1$ s are presented in Table II. Local

DOI: 10.1002/fld

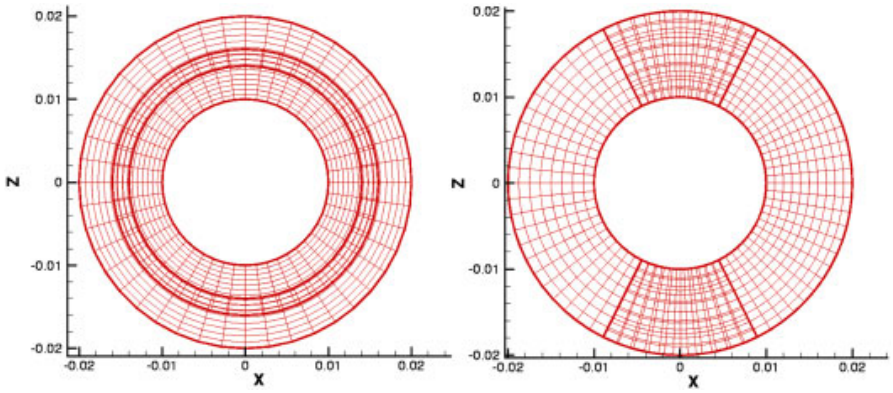


Figure 9. Block-structured curvilinear meshes.

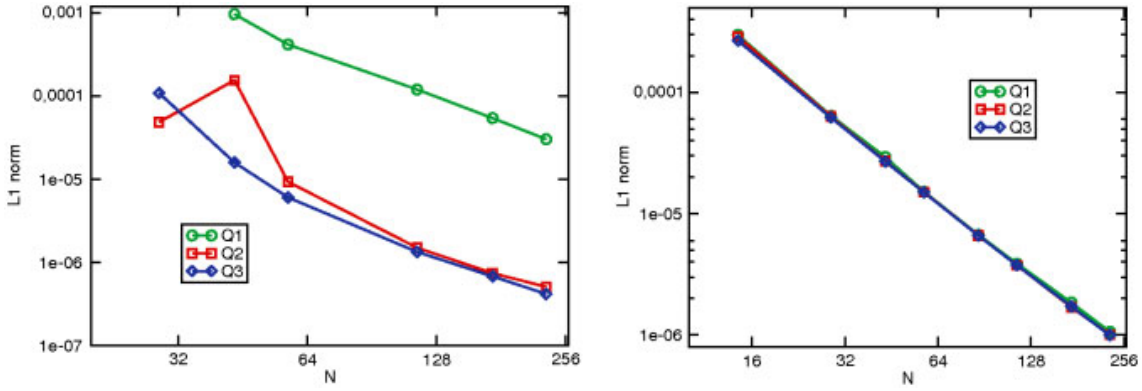


Figure 10. Convergence order for radial flow; radial (left) and polar (right) sections.

slope in a log/log scale gives a time convergence order of one, coherent with the scheme used (Euler time scheme and linearization at order 1). With very small time step, the stagnation of the error is due to the saturation of the time error by the spatial one.

4.3. Couette and radial flows (curvilinear meshes)

In order to validate the method on 2D curvilinear meshes, we studied the spatial convergence order on two academic cases (Couette and radial flows) based upon two structured non-conforming blocks (see Figure 9). The interfaces can be along the radius or at a constant angle (radial or polar section).

Results are presented for each type of section and for Q^1, Q^2 and Q^3 interpolation polynomials. The L_1 norm of the error compared to the analytical solution was chosen to represent the order of convergence.

As shown in Figures 10 and 11, the spatial convergence order is 2 whatever the polynomial used. We note that the level of errors obtained with the Q^1 polynomial is higher than for the Q^2 and Q^3 polynomials.

DOI: 10.1002/fld

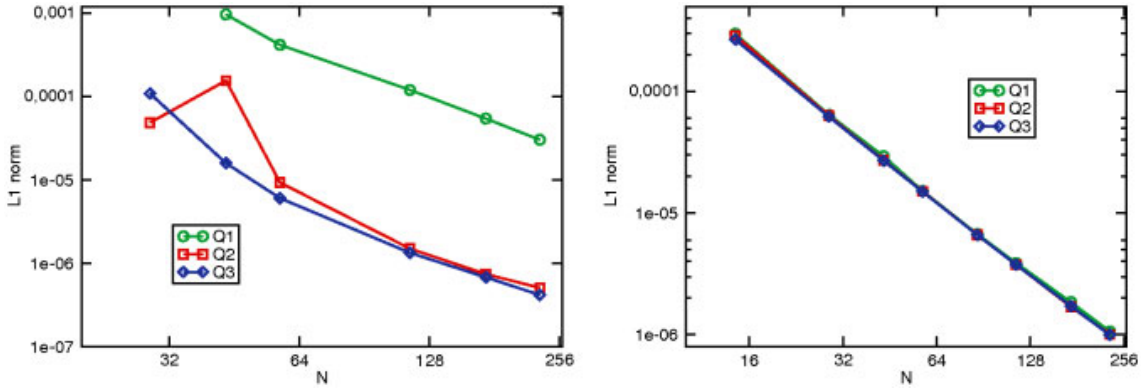


Figure 11. Convergence order for Couette flow; radial (left) and polar (right) cutting.

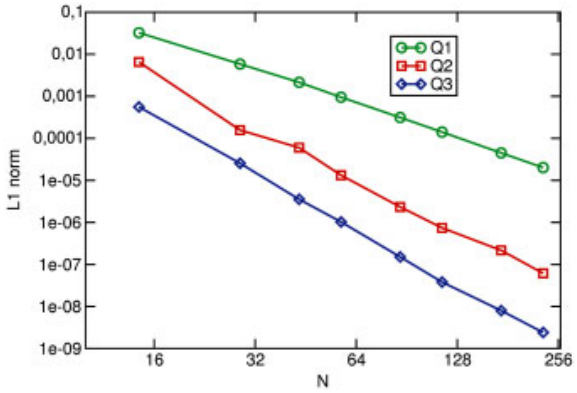


Figure 12. L_1 norm of the velocity divergence for radial flow (polar section).

4.4. *Conclusion on the convergence order of the method*

Generally speaking, it can be observed that the spatial convergence order of the code is not deteriorated by the interpolation. The Q^1 polynomial is not really interesting because error levels are too high. The Q^3 polynomial is not of a great interest: error levels are slightly lower than with the Q^2 polynomial but they need many more points to interpolate. Moreover, velocity divergence levels are also dependent on the order of interpolation (see Figure 12). For the further simulations, we chose the Q^2 polynomial because it was a good compromise between error level, velocity divergence and ease of implementation.

5. NUMERICAL SIMULATIONS

5.1. *Driven cavity* ($Re = 1000$)

5.1.1. Description of the test case. Since the early work of Burggraf [35], lid-driven cavity flow has been considered as a classic test problem for the assessment of numerical methods and the

DOI: 10.1002/fld

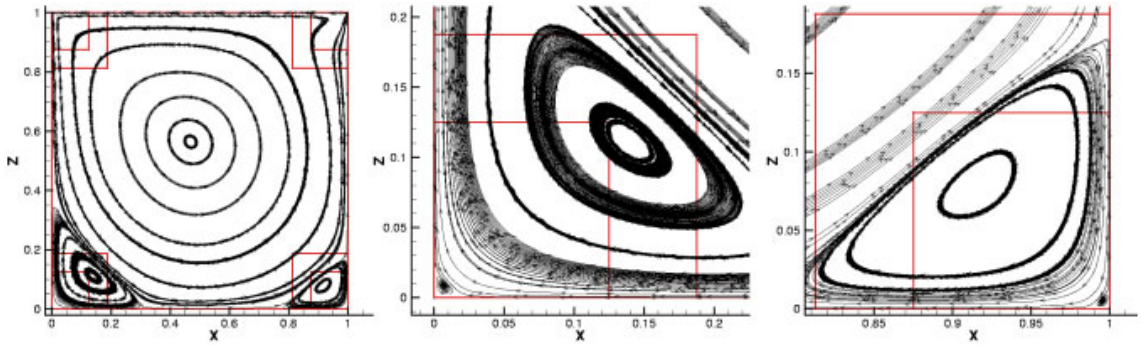


Figure 13. Streamlines of the driven cavity flow.

Table III. Comparison of the intensity and the position (x, y) of the main vortex.

Reference	Mesh size	Maximum streamline	X	Y
Aq. block-structured	121 344 ($\approx 348^2$)	0.11877	0.4687	0.5664
Aq. monoblock	512^2	0.11893	0.4693	0.5642
Botella and Peyret	512^2	0.11894	0.4692	0.5652
Bruneau	256^2	0.1163	0.4687	0.5586
Barragy *et al.*	256^2	0.11893	—	—
Schreiber *et al.*	141^2	0.11603	0.4714	0.5643
Ghia *et al.*	128^2	0.11793	0.4687	0.5625

validation of CFD codes. In the present article, we refer to the work of Botella and Peyret [36]. Their article is a benchmark rich in References [37–40], for which the main results are given for the singular driven cavity flow at a Reynolds number of 1000. The upper boundary tangential velocity leads to the formation of a main vortex that fills the main part of the cavity. In the lower corners, secondary and ternary vortices appear (see Figure 13).

The mesh is divided into five structured blocks (see Figure 13): the first block occupies the main part of the domain; the two lower (upper) blocks have step spaces a third (half) the size of the main block. There are 121 344 elements ($\approx 256 \times 256$ on the main block, 64×64 for an upper block and 96×96 for a lower one). In this case, the block-structured mesh is used to refine the description of the flow in the corners of the domain.

The main results found in the literature are given on the intensity and position of the vortices. Our results are obtained with convergence criteria on stationarity below 10^{-12}. We also present results obtained on a monoblock mesh with a Chebychev polynomial step size variation.

5.1.1.1. Position and intensity of the vortices. The results presented in Tables III–VII show a good correlation between our values and those of other authors.

5.1.1.2. Velocity and pressure profile. Figures 14 and 15 show comparisons of the velocity profiles between monoblock, block-structured and those obtained by Bottela and Peyret [36].

DOI: 10.1002/fld

Table IV. Comparison of the intensity and the position (x, y) of the left secondary vortex.

Reference	Mesh size	Maximum streamline	X	Y
Aq. block-structured	121 344 ($\approx 348^2$)	-1.7285×10^{-3}	0.1354	0.1120
Aq. monoblock	512^2	-1.7314×10^{-3}	0.1358	0.1116
Botella and Peyret	512^2	-1.7297×10^{-3}	0.1360	0.1118
Bruneau	256^2	-1.91×10^{-3}	0.1289	0.1094
Schreiber *et al.*	141^2	-1.7×10^{-3}	0.1357	0.1071
Ghia *et al.*	128^2	-1.7210×10^{-3}	0.1406	0.1094

Table V. Comparison of the intensity and the position (x, y) of the right secondary vortex.

Reference	Mesh size	Maximum streamline	X	Y
Aq. block-structured	121 344 ($\approx 348^2$)	-2.3481×10^{-4}	0.9167	0.0781
Aq. monoblock	512^2	-2.3347×10^{-4}	0.9157	0.0776
Botella and Peyret	512^2	-2.3345×10^{-4}	0.9167	0.0781
Bruneau	256^2	-3.25×10^{-4}	0.9141	0.0820
Schreiber *et al.*	141^2	-2.17×10^{-4}	0.9143	0.0714
Ghia *et al.*	128^2	-2.3113×10^{-4}	0.9141	0.0781

Table VI. Comparison of the intensity and the position (x, y) of the left ternary vortex.

Reference	Mesh size	Maximum streamline	X	Y
Aq. block-structured	121 344 ($\approx 348^2$)	-4.7314×10^{-8}	0.00781	0.00781
Aq. monoblock	512^2	-5.0239×10^{-8}	0.00789	0.00736
Botella and Peyret	512^2	-5.0399×10^{-8}	0.00768	0.00765
Ghia *et al.*	128^2	-9.3193×10^{-8}	0.0078	0.0078

Table VII. Comparison of the intensity and the position (x, y) of the right ternary vortex.

Reference	Mesh size	Maximum streamline	X	Y
Aq. block-structured	121 344 ($\approx 348^2$)	—	0.9952	0.00463
Aq. monoblock	512^2	6.3462×10^{-9}	0.9950	0.00497
Botella and Peyret	512^2	6.33255×10^{-9}	0.9951	0.00482
Bruneau	128^2	3.06×10^{-9}	0.9961	0.0039

Small differences on velocity profiles between the monoblock and block-structured solutions can be explained by grid resolution differences. As regards the pressure profiles (see Figure 16), good agreement can be observed with those obtained with a monoblock mesh, except on the interface of the upper blocks. The overpressure (depression) observed in the upper right (upper left) block is the result of the velocity divergence levels, which are constant in each block (around 10^{-5}). We did not observe these phenomena on other test cases presented in this paper where pressure

DOI: 10.1002/fld

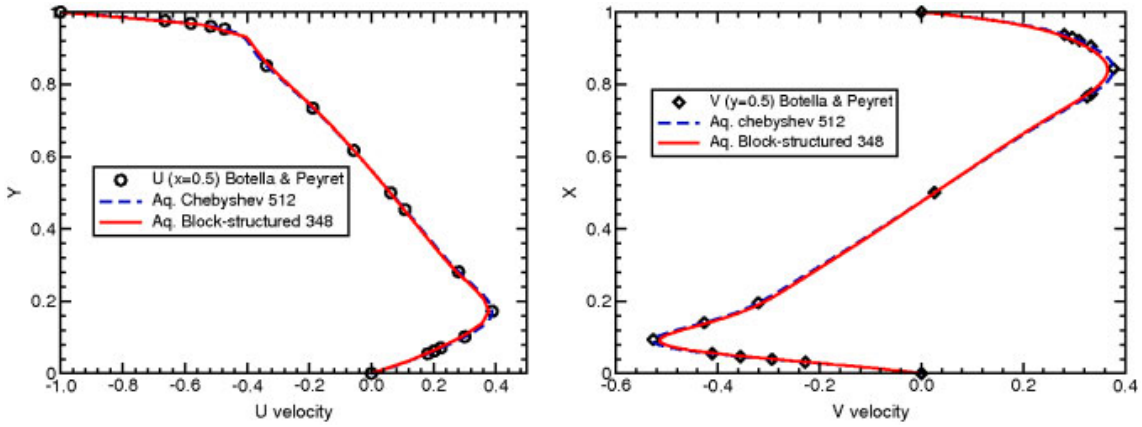


Figure 14. U profiles (left) at $X = 0.5$ and V profiles (right) at $Y = 0.5$.

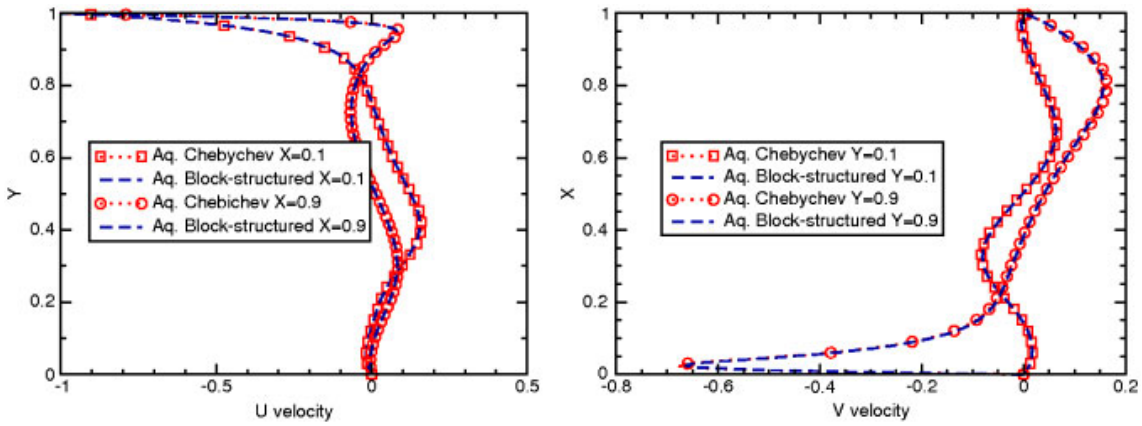


Figure 15. U profile (left) at $X = 0.1$ and 0.9. V profile (right) at $Y = 0.1$ and 0.9.

was continuous through the interfaces. There is no doubt that a Neumann condition relaxes the constraint through the different blocks. When the domain is closed, a lack of conservation of mass and momentum due to interpolation explain the pressure shift observed.

5.2. *Flow around a backward facing step* ($Re \leqslant 1000$)

5.2.1. Description of the test case. The backward-facing step is one of the most fundamental examples where laminar separation is caused by the sudden change in the geometry. The extension of the section induces a reverse pressure gradient that leads to a separation of the flow into several zones: a recirculation appears behind the step and, when the Reynolds number increases, a second recirculation appears on the upper wall. The experimental reference works are those of Armaly *et al.* [41] and, to a lesser degree, the work of Lee and Matescu [42]. The Reynolds number, based on the height of the channel H_d, the bulk velocity at the entrance of the channel U_{moy} and the kinematic viscosity, vary between 100 and 1000.

DOI: 10.1002/fld

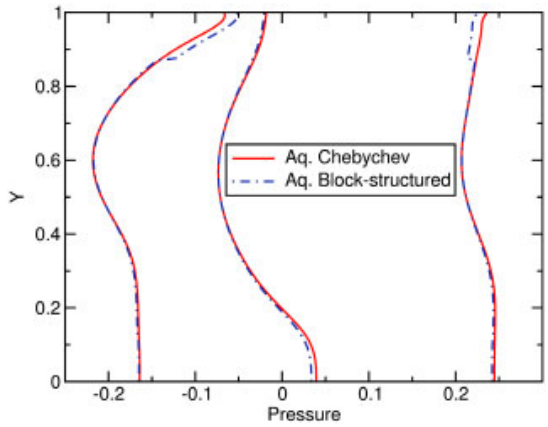


Figure 16. Pressure profiles at $X = 0.1$, 0.5 and 0.9.

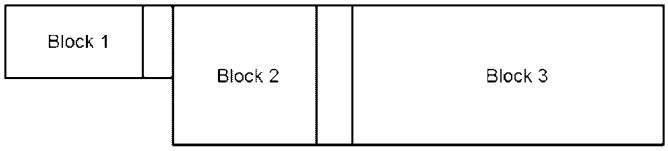


Figure 17. Block-structured mesh for the backward-facing step flow.

Armaly's experiments were performed on a channel of width equal to 36.7 times the step height. Flow can be considered as 2D below $Re = 400$, wall effects being not negligible for higher Reynolds numbers. Comparisons between the experiments and the results of other authors were made on the detachment and reattachment positions.

In the present work, we used a monoblock mesh and a block-structured one (see Figure 17): the domain was divided into three blocks, the middle one being twice as fine as the others. The interface between block 2 and block 3 was positioned at the middle of the recirculation, at a distance of 6.5 m from the step foot (see Figure 18).

We will compare the results obtained using two non-conforming meshes (3456 and 24 576 elements) with those obtained with a monoblock mesh and those of several other authors. The stationarity criterion was set to 10^{-10} whatever the Reynolds number. We have separated the results in two parts: for $Re \leqslant 400$ and for $Re > 400$. In the first case, there is no upper recirculation and 3D effects are negligible.

5.2.1.1. Lower recirculation $Re \leqslant 400$. Agreement with the experiments and various numerical results [15, 43–45] is very good as shown in Figure 19.

These results were obtained without the vectorial projection method. Velocity divergence is null over the whole domain except at the interface node where it is around 10^{-4} due to the non-conservative interpolation. We performed other simulations with the vectorial projection method. Therefore, divergence is null on the interface without any strong influence on the recirculation length as shown in Table VIII.

DOI: 10.1002/fld

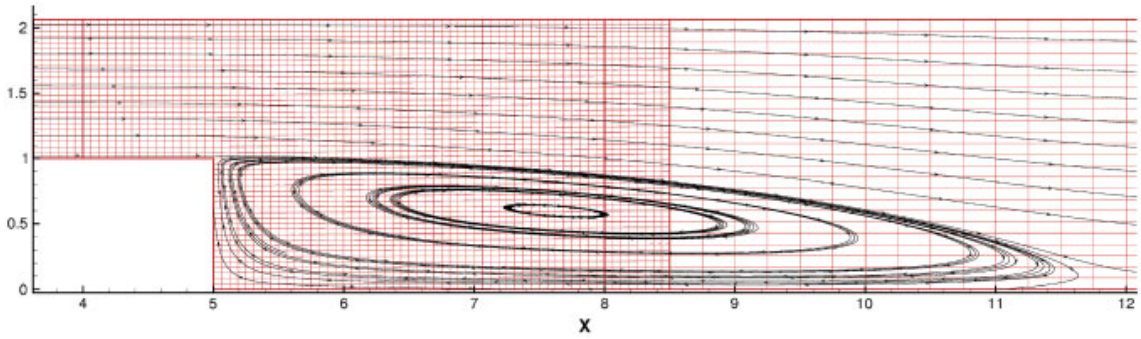


Figure 18. Streamlines through the interfaces ($Re = 500$).

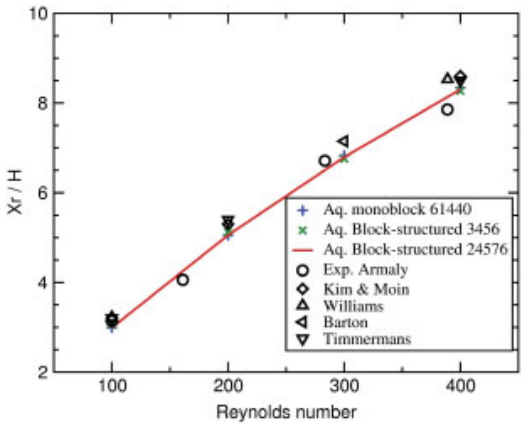


Figure 19. Recirculation length (X_r/H) for $Re \leqslant 400$. X_r is the length of the lower recirculation and H the step height.

Table VIII. Influence of the vectorial projection method on the recirculation length ($Re \leqslant 400$).

Re	100	200	300	400
Recirculation length	2.99	5.06	6.79	8.30
Correction in %	4.3×10^{-2}	2.3×10^{-1}	4.2×10^{-2}	5.1×10^{-2}

5.2.1.2. Lower recirculation for Re>400. The upper recirculation appears and the reattachment point of the lower recirculation continues to move off the foot of the step. The experimental results of Armaly and Lee are quite different (see Figure 20). If we compare our results to those of Armaly, the length of the lower recirculation is underestimated with an error of 8.6% by the monoblock mesh and 9% by the block-structured one at $Re = 500$. At $Re = 1000$, errors are around 19%. Williams and Baker [45] have shown with a 3D simulation that these differences are due to 3D effects. Indeed, most other 2D numerical studies found in the literature [43–47] show the same order of error (see Figure 20 and Table IX).

DOI: 10.1002/fld

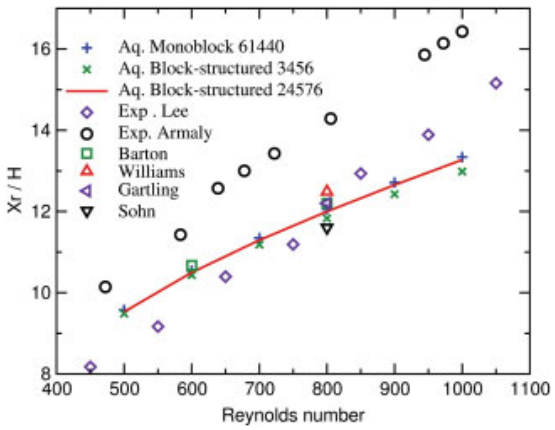


Figure 20. Lower recirculation length (X_r/H) for $Re>400$.

Table IX. Lower recirculation length, $Re=800$.

	Lee	Armaly	Aq. monoblock	Aq. block-structured	Lee	Gartling	Kim	Sohn
X_r	12.9	14.2	12.05	11.99	12	12.2	12	11.6

Table X. Influence of the correction on the lower recirculation length ($Re>400$).

Re	500	600	700	800	900	1000
Recirculation length	9.53	10.50	11.29	12	12.65	13.28
Correction in %	4.2×10^{-2}	7.6×10^{-3}	4.4×10^{-3}	1.7×10^{-2}	1.9×10^{-2}	2.4×10^{-2}

Finally, Table X shows very low differences if we use the vectorial projection method to reach null divergence over each block.

5.2.1.3. Upper recirculation for Re>400. The abscissa of the detachment point on the upper wall is underestimated, from $Re=600$, compared to the results of Armaly (see Figure 21 left). Compared to the Lee's results, it is overestimated until $Re=750$ and then underestimated. Again, 3D effects explain these differences. They are less obvious if one looks at the abscissa of the upper reattachment point (see Figure 21 right).

It must be noted that the length of the upper recirculation at $Re=800$ is overestimated by around 18% with a monoblock mesh and by 15% with a block-structured one. As noted by Armaly, the lengths of the lower and the upper recirculations are strongly coupled, an underestimation of one of them involving an overestimation of the other. Our results are in agreement with other numerical studies found in the literature (see Table XI).

The flow rate difference between the entrance and the exit of the domain is around $6.10^{-4}\,m^3\,s^{-1}$ whatever the Reynolds number. The non-conservative interpolation finally has little influence on the accuracy of the results. Velocity divergence on the interface between the blocks is relatively small, and decreases with mesh refinement as shown in Figure 22.

DOI: 10.1002/fld

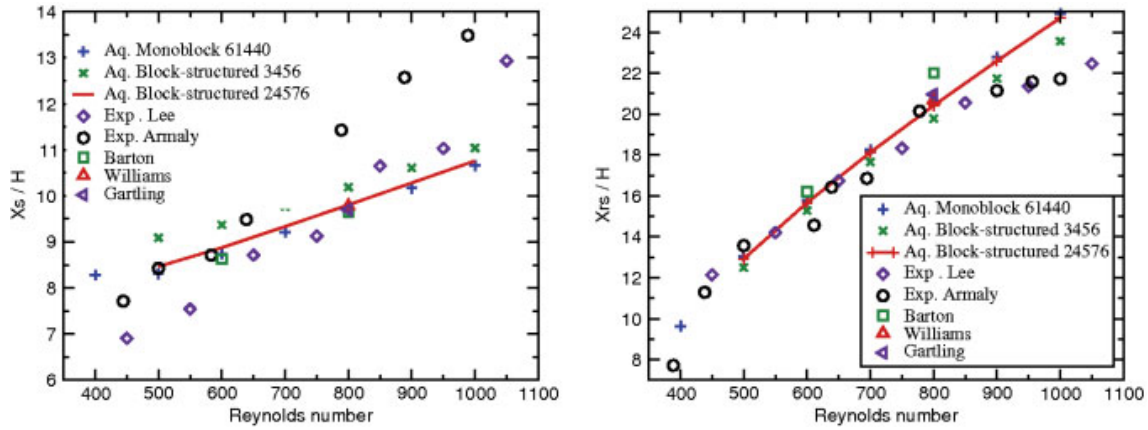


Figure 21. Abscissae of the detachment (X_s) and reattachment (X_{rs}) points *versus* Reynolds number.

Table XI. Comparison of the abscissae of the detachment and reattachment points at $Re = 800$.

	Exo. Lee	Exp. Armaly	Aq. monoblock	Aq. block-structured	Lee	Gartling	Kim	Sohn
X_S	10.3	11.5	9.69	9.80	9.6	9.7	—	—
X_{rs}	19.5	20	20.57	20.41	20.6	20.96	—	—
$X_{rs} - X_s$	9.2	8.5	10.88	10.61	11	11.26	11.5	9.26

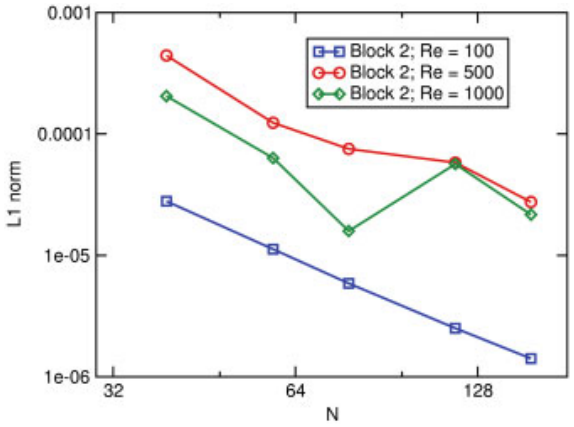


Figure 22. Mean divergence on the block interface as a function of the mesh size for different Reynolds numbers.

Figure 23 shows several velocity profiles and a good agreement between the monoblock and the block-structured solutions. The non-conservative interpolation does not produce significant differences with conforming meshes.

DOI: 10.1002/fld

5.3. *Flow around a cylinder*

The flow around a cylinder has been widely studied from the theoretical, experimental and numerical points of view. The problem can be characterized by the Reynolds number, based on the diameter of the cylinder and the entrance velocity. We are interested in two flow ranges:

- $4<Re<49$: the inertial term of the momentum is not negligible. One can observe a detachment of the streamlines around the cylinder and the presence of two symmetrical, stable vortices, the lengths which increase with the Reynolds number.
- $49<Re<190$: the flow is not stationary. The result is a Von Karman alley with alternating vortex detachment, the frequency of which increases with the Reynolds numbers characterized by the Strouhal number.

5.3.1. Parameters of the test case. To validate our method, we studied the length of the recirculation; the detachment angle and the drag coefficient C_x for Reynolds numbers below 49. We compared our results with the experimental work of Taneda [48], Tritton [49], Acrivos *et al*. [50], Coutanceau and Bouard [51], and with the numerical works of Dennis and Chang [52], Nieuwstadt and Keller [53], Fornberg [54] and, more recently, He and Doolen [55]. For higher Reynolds number, we compared our results on the Strouhal number to the law of Williamson and Brovon [56], which is in good agreement with experiments.

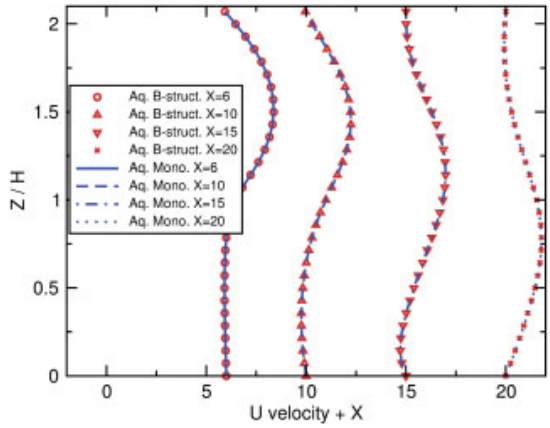


Figure 23. u component profiles for monoblock and block-structured meshes at $Re = 800$.

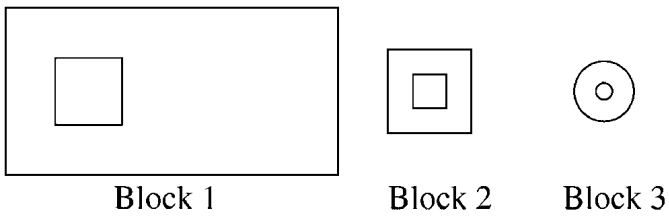


Figure 24. Block-structured mesh composition for the study of the flow around a cylinder.

DOI: 10.1002/fld

Table XII. Recirculation length, detachment angle and drag coefficient at $Re = 10$.

Authors	L/r	θ_d	C_x
Aq. block-structured	0.549	25.47	3.077
Dennis and Chang [52]	0.53	29.6	2.846
Nieuwstadt and Keller [53]	0.434	27.96	2.828
Coutanceau and Bouard [51]	0.68	32.5	—
He and Doolen [55]	0.474	26.89	3.170

Table XIII. Recirculation length, detachment angle and drag coefficient at $Re = 20$.

Authors	L/r	θ_d	C_x
Aq. block-structured	2.16	41.98	2.070
Dennis and Chang [52]	1.88	43.7	2.045
Nieuwstadt and Keller [53]	1.786	43.37	2.053
Coutanceau and Bouard [51]	1.86	44.8	—
Fornberg [54]	1.82	—	2.000
He and Doolen [55]	1.842	42.96	2.152

Table XIV. Recirculation length, detachment angle and drag coefficient at $Re = 30$.

Authors	L/r	θ_d	C_x
Aq. block-structured	3.57	47.93	1.702
Nieuwstadt and Keller [53]	3.09	—	1.716

Table XV. Recirculation length, detachment angle and drag coefficient at $Re = 40$.

Authors	L/r	θ_d	C_x
Aq. block-structured	4.94	50.99	1.503
Dennis and Chang [52]	4.69	53.8	1.522
Nieuwstadt and Keller [53]	4.357	53.34	1.550
Coutanceau and Bouard [51]	4.26	53.5	—
Fornberg [54]	4.48	—	1.498
He and Doolen [55]	4.490	52.84	1.499
Prabhakar [57]	4.55	—	1.55

Figure 24 shows the block-structured mesh composition we used. It is based on the superposition of three blocks: the first two are Cartesian blocks with a hole, the last one is a disk. Space step size becomes increasingly thin. The total number of elements is 33 168. Around the cylinder of diameter 1, there are 240 elements according to θ, and space step size in the r direction is 0.25 m.

5.3.2. Stationary flow $4<Re<49$. As shown in Tables XII–XV, the recirculation length is generally overestimated compared to other numerical results found in the literature. The error reaches 26%

DOI: 10.1002/fld

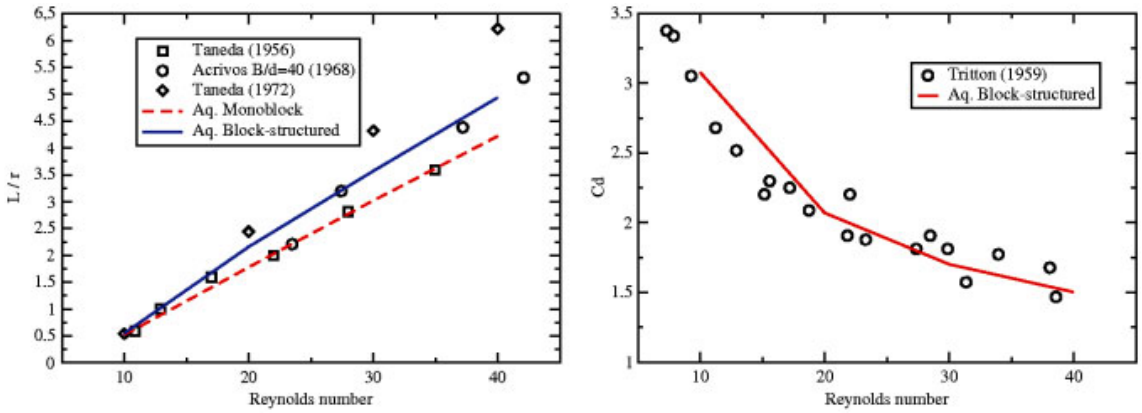


Figure 25. Recirculation length (left) and drag coefficient (right) *versus* the Reynolds number.

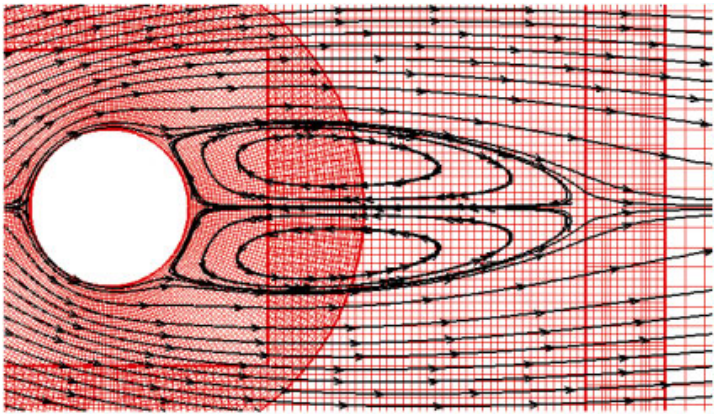


Figure 26. Streamlines of the flow around a cylinder at $Re = 40$.

Table XVI. Strouhal number at various Reynolds numbers.

Re	Block-structured	Williamson
80	0.158	0.155
100	0.170	0.166
120	0.180	0.175
140	0.185	0.182
160	0.192	0.188

with respect to Nieuwstadt and Keller [53] at $Re = 10$. Differences decrease with the increase of the Reynolds number (around 10% at $Re = 40$). Nevertheless, our values are in the uncertainty area of the experiment (see Figure 25 left). The same comments can be made regarding the detachment angle. Results obtained on the drag coefficient (see Figure 25 right) are much closer to those found in the literature. Figure 26 shows the mesh and the streamlines around the cylinder and through the interfaces.

DOI: 10.1002/fld

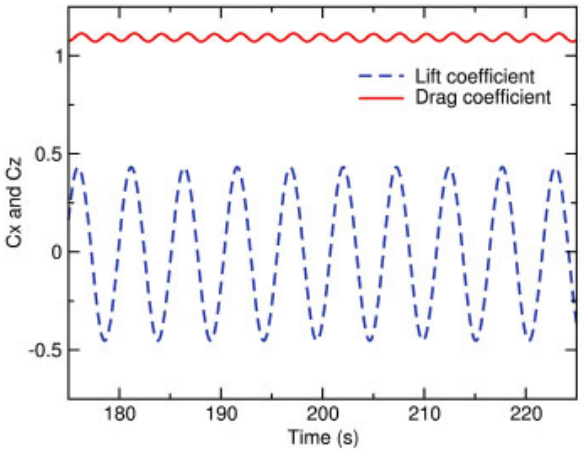


Figure 27. Drag and lift coefficients at $Re = 160$.

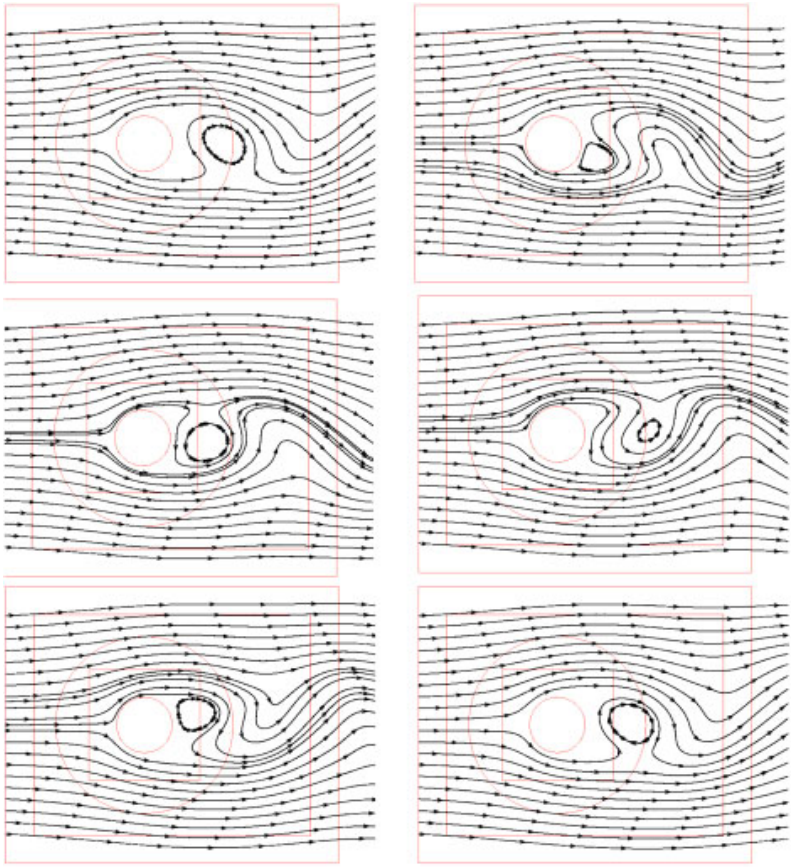


Figure 28. Streamlines of the flow around a cylinder during a period at $Re = 160$.

DOI: 10.1002/fld

5.3.3. Unstationary flow 49<Re<190. As shown in Table XVI, results are very close to the Williamson Strouhal law. The overestimation is around 3%. At $Re = 160$, Figures 27 and 28 show the time variation of the lift and drag coefficients and the streamlines around the cylinder during a period.

6. CONCLUSION

In the present work, we have proposed and validated a connecting technique for non-conforming and overlapping block-structured meshes. It is non-iterative and based on an implicit non-conservative interpolation. This is performed in the context of the augmented Lagrangian method and the iterative Uzawa algorithm. The linear system of the Navier–Stokes equations is modified: lines that correspond to the interface nodes link the blocks together. Thus, the discretization of the equations and the connecting conditions are implicit.

With a series of test cases for which the analytical solution is known, we have demonstrated that the spatial convergence order is 2. Numerical simulations of laminar and incompressible flows show the feasibility and validity of the method.

Nevertheless, the interpolation used does not allow the incompressibility constraint to be fully attained over the whole domain. The problem of mass conservation through the interface remains. This problem is altogether logical because the interpolation acts as a Dirichlet condition on the interfaces.

We have modified the vectorial projection method to ensure mass conservation locally to each block. New conditions on the interfaces have been written that take into account the flow rate difference at each interface between blocks. In particular cases, it could be possible to take into account the flow rate difference between blocks or to bring each block flow rate to a reference one (to the entrance flow rate for instance). In the future, it will be necessary to connect gradient-type information between the blocks (normal constraint) to ensure a better conservation of the momentum and the incompressible constraint. The main difficulty to be dealt with lies in the implicit treatment of this new condition.

ACKNOWLEDGEMENTS

We acknowledge the reviewers for their valuable contributions which have been of great help in improving the quality of the presentation and discussion of the results. The authors thank Professor Mejdi Azaïez for his critical reading, helpful suggestions and fruitful discussions that guided us to achieve this work. We acknowledge calculation facilities provided by the 'M3PEC-Mésocentre Régional' main frame supercomputers of the University Bordeaux 1 financed by the University, the 'Conseil Régional d'Aquitaine' and the French Ministry of Research and Technology.

REFERENCES

1. Romé C, Glockner S. An implicit multiblock coupling for the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 2005; **47**:1261–1267.
2. Khadra K, Parneix S, Angot P, Caltagirone J-P. Fictitious domain approach for numerical modelling of Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 2000; **34**:651–684.
3. Peyret R, Taylor TD. *Computational Methods for Fluid Flow*. Springer: New York, 1983.
4. Girault V, Raviart P-A. *Finite Element Methods for Navier–Stokes Equations*. Springer Series in Computational Mathematics. Springer: New York, 1986.

DOI: 10.1002/fld

5. Fortin M, Glowinski R. Méthodes de Lagrangien Augmenté, Application à la résolution numérique de problèmes aux limites. *Collection Méthodes Mathématiques de l'Informatique*, Dunod, 1982.
6. Vincent S, Caltagirone J-P, Lubin P, Randrianarivelo TN. An adaptative augmented Lagrangian method for three-dimensional multimaterial flows. *Computers and Fluids* 2004; **33**:1273–1289.
7. Breil J, Caltagirone JP. Three dimensional computer simulation of mould filling with N fluids by VOF PLIC and projection methods. *ICCFD*, Kyoto, 10–14 July 2000.
8. Le Bot C, Vincent S, Arquis E. Impact and solidification of indium droplets on a dry substrate. *International Journal of Thermal Sciences* 2005; **44**:219–233.
9. Lubin P, Vincent S, Caltagirone JP, Abadie S. Fully three-dimensional direct numerical simulation of a plunging breaker. *Comptes Rendus - Mecanique* 2003; **331**(7):495–501.
10. Chorin AJ. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics* 1967; **2**:12–26.
11. Chorin AJ. Numerical simulation of the Navier–Stokes equations. *Mathematics of Computation* 1968; **22**:745–762.
12. Gresho PM, Chan ST. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part I: Theory. Part II: Implementation. *International Journal for Numerical Methods in Fluids* 1990; **11**:587–620, 621–659.
13. Hugues S, Randriamampianina A. An improved projection scheme applied to pseudospectral methods for the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 1998; **28**:501–521.
14. Goda K. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal of Computational Physics* 1978; **30**:76–95.
15. Timmermans LJP, Minev PD, Van de Vosse FN. An approximate projection scheme for incompressible flow using spectral elements. *International Journal for Numerical Methods in Fluids* 1996; **22**:673–688.
16. Patankar SV, Spalding D. A calculation procedure for heat mass and momentum transfer in three dimensional parabolic flows. *International Journal of Heat and Mass Transfer* 1972; **15**:1787–1806.
17. Patankar SV. *Numerical Heat Transfer and Fluid flow*. Hemisphere Publishing Corporation: New York, 1980.
18. Caltagirone JP, Breil J. Sur une méthode de projection vectorielle pour la résolution des équations de Navier–Stokes. *Comptes Rendus de l' Academie des Sciences*, *Paris*, *Serie IIb* 1999; **327**:1179–1184.
19. Harlow FH, Welsh JE. Numerical calculation of time dependent viscous incompressible flows. *The Physics of Fluids* 1965; **8**:2182–2189.
20. Amestoy PR, Duff IS, L'Excellent J-Y. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering* 2000; **184**:501–520.
21. Quarteroni A, Valli A. *Domain Decomposition Methods for Partial Differential Equations*. *Numerical Mathematics and Scientific Computation*. Oxford Science Publications: Oxford, 1999.
22. Lions PL. On the Schwarz alternating method III: a variant for nonoverlapping subdomains. *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1990.
23. Bernardi C, Maday Y, Patera AT. Domain decomposition by the mortar element method. In *Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters*. Kaper HG, Garbey M (eds). NATO ASI Series C, vol. 384. Kluwer Academic Publishers: Dordrecht, 1993; 269–286.
24. Cai XC, Dryja M, Sarkis M. Overlapping nonmatching grid Mortar element methods for elliptic problems. *SIAM Journal on Numerical Analysis* 1999; **36**:581–606.
25. Achdou Y, Maday Y, Widlund OB. Iterative substructuring preconditioners for mortar element methods in two dimensions. *SIAM Journal on Numerical Analysis* 1999; **36**(2):551–580.
26. Achdou Y, Japhet C, Maday Y, Nataf F. A new cement to glue non-conforming grids with Robin interface conditions: the finite volume case. *Numerische Mathematik* 2002; **92**(4):593–620.
27. Arbogast T, Yotov I. A non-mortar mixed finite element method for elliptic problems on non-matching multiblock grids. *Computer Methods in Applied Mechanics and Engineering* 1997; **149**(1–4):255–265.
28. Benek JA, Buning PG, Steger JL. A 3-D Chimera grid embedding technique. *AIAA 85-1523CP*, 1985.
29. Steger JL, Benek JA. On the use of composite grid schemes in computational aerodynamics. *Computer Methods in Applied Mechanics and Engineering* 1987; **64**:301–320.
30. Houzeaux G, Codina R. A Chimera method based on a Dirichlet/Neumann (Robin) coupling for the Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 2003; **192**:3343–3377.
31. Brezzi F, Lions JL, Pironneau O. Analysis of a Chimera method. *Comptes Rendus de l' Academie des Sciences*, *Paris*, *Série I* 2001; **332**(I):655–660.

DOI: 10.1002/fld

32. Cadafalch J, Oliva A, Pérez-Segarra CD, Costa M, Salom J. Comparative study of conservative and nonconservative interpolation schemes for the domain decomposition method on laminar incompressible flow. *Numerical Heat Transfer*, *Part B* 1999; **35**:65–84.
33. Henshaw WD, Watson TJ. A fourth-order accurate method for the incompressible Navier–Stokes equations on overlapping grids. *Journal of Computational Physics* 1994; **113**(1):13–25.
34. Meakin RL. On the spatial and temporal accuracy of overset grid methods for moving body problems. *AIAA Applied Aerodynamics Conference*, 12th, Colorado Springs, CO, 20–22 June 1994, *Technical Papers*, Pt. 2 (A94-30939 10-02), American Institute of Aeronautics and Astronautics, Washington, DC, 1994; 858–871.
35. Burggraf OR. Analytical and numerical studies of the structure of the steady separated flow. *Journal of Fluid Mechanics* 1966; **24**:113–151.
36. Botella O, Peyret R. Benchmark spectral results on the lid-driven cavity flow. *Computers and Fluids* 1998; **4**:421–433.
37. Ghia U, Ghia KN, Shin CT. High-resolutions for incompressible flow using the Navier–Stokes equations and a multi-grid method. *Journal of Computational Physics* 1982; **48**:387–411.
38. Schreiber R, Keller HB. Driven cavity flow by efficient numerical techniques. *Journal of Computational Physics* 1983; **49**:310–333.
39. Bruneau CH, Jouron C. An efficient scheme for solving steady incompressible Navier–Stokes equations. *Journal of Computational Physics* 1990; **89**:389–413.
40. Barragy E, Carey GF. Stream function–vorticity driven cavity solution using p finite elements. *Computers and Fluids* 1997; **26**:453–468.
41. Armaly PH, Durst F, Pereira JCF, Schonung F. Experimental and theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics* 1983; **127**:473–496.
42. Lee T, Matescu D. Experimental and numerical investigation of 2-D backward-facing step flow. *Journal of Fluids and Structures* 1988; **12**:703–717.
43. Kim J, Moin P. Application of a fractional-step method to incompressible Navier–Stokes equations. *Journal of Computational Physics* 1985; **59**:308–323.
44. Barton IE. A numerical study of flow over a confined backward-facing step. *International Journal for Numerical Methods in Fluids* 1995; **21**:653–665.
45. Williams PT, Baker AJ. Numerical simulations of laminar flow over a 3D backward-facing step. *International Journal for Numerical Methods in Fluids* 1997; **24**:1159–1183.
46. Gartling DK. A test problem for outflow boundary conditions—flow over a backward-facing step. *International Journal for Numerical Methods in Fluids* 1990; **11**:953–967.
47. Sohn J. Evaluation of FIDAP on some classical laminar and turbulent benchmarks. *International Journal for Numerical Methods in Fluids* 1988; **8**:1469–1490.
48. Taneda S. Experimental investigation of the wakes behind cylinder at low Reynolds numbers. *Journal of the Physical Society of Japan* 1956; **11**:302–306.
49. Tritton DJ. Experiments on the flow past a circular cylinder at low Reynolds numbers. *Journal of Fluid Mechanics* 1959; **6**:547–567.
50. Acrivos A, Leal LG, Snowden DD, Pan F. Further experiments on steady separated flows past bluff objects. *Journal of Fluid Mechanics* 1968; **34**:25–48.
51. Coutanceau M, Bouard R. Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow. *Journal of Fluid Mechanics* 1977; **79**:231–256.
52. Dennis SCR, Chang GZ. Numerical solutions for steady viscous flow past a circular cylinder. *Journal of Fluid Mechanics* 1970; **42**:471–489.
53. Nieuwstadt F, Keller HB. Viscous flow past circular cylinders. *Computer and Fluids* 1973; **1**:59.
54. Fornberg B. A numerical study of steady viscous flow past a circular cylinder. *Journal of Fluid Mechanics* 1980; **98**:819–855.
55. He X, Doolen G. Lattice Boltzman method on curvilinear coordinates system: flow around a circular cylinder. *Journal of Computational Physics* 1997; **134**:306–315.
56. Williamson CHK, Brown GL. A series in $1/\sqrt{Re}$ to represent the Strouhal–Reynolds number relationship of the cylinder wake. *Journal of Fluids and Structures* 1998; **12**:1073–1085.
57. Prabhakar V, Reddy JN. Spectral/hp penalty least-squares finite element formulation for the steady incompressible Navier–Stokes equations. *Journal of Computational Physics* 2006; **215**(1):274–297.

DOI: 10.1002/fld

3 An implicit method for the Navier-Stokes equations on overlapping block-structured grids

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2010; **62**:784–801
Published online 8 April 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.2044

An implicit method for the Navier–Stokes equations on overlapping block-structured grids

E. Ahusborde*,† and S. Glockner

Laboratoire TREFLE, Université de Bordeaux, CNRS-UMR 8508, 16 av. Pey-Berland, 33607 Pessac Cedex, France

SUMMARY

This paper deals with a method first introduced by Romé *et al.* in two articles. The authors reported that their method was suitable to run the Navier–Stokes equations efficiently on non-matching and overlapping block-structured meshes. However, there was a problem of mass conservation and a discontinuity of pressure through the interfaces in some cases. In the present paper, an improvement of the method based on a pressure correction scheme is proposed. With this improvement, the pressure is continuous through the interfaces and the incompressibility constraint is ensured over the whole domain. Several numerical tests were carried out to assess the proposed method. Copyright © 2009 John Wiley & Sons, Ltd.

Received 5 November 2008; Revised 13 February 2009; Accepted 13 February 2009

KEY WORDS: block-structured meshes; Navier–Stokes; non-conforming; non-matching; overlapping; interpolation

1. INTRODUCTION

In computational fluid dynamics when flows are calculated for complex geometries, one can either use a block-structured grid or an unstructured one. Unstructured grids allow very complex geometries to be meshed leading to complex discretization schemes and solvers that require tables of connectivity between nodes and indirect addressing. If the geometry is not too complicated, it can be divided into a reasonable number of structured blocks. Lexical numbering facilitates the discretization of the equations (specially if the grid remains orthogonal) and the use of many solvers dedicated to the structured grids.

*Correspondence to: E. Ahusborde, Laboratoire TREFLE, Université de Bordeaux, CNRS-UMR 8508, 16 av. Pey-Berland, 33607 Pessac Cedex, France.
†E-mail: ahusborde@enscpb.fr

Contract/grant sponsor: Conseil Régional d'Aquitaine
Contract/grant sponsor: French Ministry of Science and Technology

Domain decomposition methods are well suited to these issues. They can be classified according to several criteria [1]. For instance, the block-structured grids can be overlapping or non-overlapping. Generally, each block is computed separately and provides the boundary conditions for the neighbouring blocks. Historically, these methods have been introduced by Schwarz [2]. The main drawback of the method is that overlapping is required for convergence. An improvement consists in substituting overlapping by another boundary condition. In [3], Lions proposed the use of a Robin boundary condition. Our strategy consists in working with overlapping in order to deal with orthogonal grids. The main difficulty is to find a relevant projection operator on the interfaces between sub-domains. Mortar element methods have been proposed to solve this problem [4, 5]. Chimera methods represent another approach [6, 7]. These methods are particularly used in aerodynamic simulations.

Non-matching meshes raise the classical question of interpolation. This difficulty becomes harder when the interpolation has to be carried out under constraint. In the case of simulation of an incompressible flow, the constraint $\nabla\cdot\mathbf{u}=0$ must be verified. Generally, interpolation is conservative if it is based on finite volume techniques [8, 9]. Fluxes through interfaces are calculated using local balance with a neighbouring block or a projection. Recently, a mass-flux-based interpolation algorithm was proposed by Tang *et al.* [10, 11]. Some authors who used non-conservative interpolation have shown that mass conservation is directly linked to the order of the interpolation [12].

In this paper, we propose an implicit method to compute the incompressible Navier–Stokes equations on block-structured meshes based on non-conservative interpolation. This study proposes an improvement of the method first introduced in [13, 14]. Indeed, the authors previously met problems to satisfy the incompressibility constraint on the interfaces between blocks leading in some cases to a discontinuity of pressure. They used the augmented Lagrangian method [15] for pressure–velocity coupling. In the present case, the method has been replaced by a pressure correction scheme [16] to circumvent these drawbacks.

We first present the numerical context of the study by describing the models and numerical methods of the CFD code Aquilon (Aq. in figures and tables). Then, we describe the novelties of the method in comparison with the method first introduced in [13, 14]. Finally, numerical tests were carried out to validate the method and clearly show the improvements.

2. NUMERICAL CONTEXT

2.1. Numerical methods

In this paper, the incompressible Navier–Stokes equations are considered:

Find the velocity $\mathbf{u}$ *and the pressure* p *such that*:

$$\nabla\cdot\mathbf{u}=0 \quad \text{in } \Omega \tag{1}$$

$$\rho\left(\frac{\partial\mathbf{u}}{\partial t}+\nabla\cdot(\mathbf{u}\otimes\mathbf{u})\right)=-\nabla p+\nabla\cdot\mu(\nabla\mathbf{u}+\nabla^t\mathbf{u}) \quad \text{in } \Omega \tag{2}$$

$$\mathbf{u}=\mathbf{0} \quad \text{on } \partial\Omega \tag{3}$$

where ρ is the density of the fluid, μ is the dynamic viscosity and $\Omega\subset\mathbb{R}^2$ is a bounded open domain with Lipschitzian border $\partial\Omega$.

DOI: 10.1002/fld

The main difficulty in the resolution of the problem (1)–(3) consists in ensuring the incompressibility constraint that couples the velocity and the pressure. In the previous study [14], the augmented Lagrangian method [15] was dealt with. In the present study, we will focus on a pressure correction scheme [16].

2.1.1. The pressure correction scheme. The pressure correction time integration scheme consists in splitting the Navier–Stokes system into two stages, a velocity prediction and a pressure correction [16]. The time interval $[0, T]$ is divided into N equidistant time steps of length $\Delta t = T/N$. The approximate velocity and pressure fields at time $t^n = n\Delta t$ $(n = 0, \ldots, N)$ are denoted $\mathbf{u}^n$ and p^n, respectively. Assuming all quantities are known up to t^n, the solution at t^{n+1} results from the velocity prediction step:

Find $\mathbf{u}_*^{n+1}$ *such that*:

$$\rho\left(\frac{\mathbf{u}_*^{n+1}-\mathbf{u}^n}{\Delta t}+\nabla\cdot(\mathbf{u}_*^{n+1}\otimes\mathbf{u}^n)-\mathbf{u}_*^{n+1}\nabla\cdot\mathbf{u}^n\right)=-\nabla p^n+\nabla\cdot\mu(\nabla\mathbf{u}_*^{n+1}+\nabla^t\mathbf{u}_*^{n+1}) \quad \text{in } \Omega \tag{4}$$

$$\mathbf{u}_*^{n+1}=\mathbf{0} \quad \text{on } \partial\Omega \tag{5}$$

followed by the pressure correction step:

Find $\mathbf{u}^{n+1}$ *and* ϕ^{n+1} *such that*:

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}_*^{n+1}}{\Delta t}+\nabla\phi^{n+1}=\mathbf{0} \quad \text{in } \Omega \tag{6}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0 \quad \text{in } \Omega \tag{7}$$

$$\mathbf{u}^{n+1}\cdot\mathbf{n}=0 \quad \text{on } \partial\Omega \tag{8}$$

with:

$$\phi^{n+1}=p^{n+1}-p^n+\mu\nabla\cdot\mathbf{u}_*^{n+1} \tag{9}$$

Considering ρ constant and taking the divergence of (6) gives:

$$\Delta\phi^{n+1}=\frac{\rho}{\Delta t}\nabla\cdot\mathbf{u}_*^{n+1} \quad \text{in } \Omega \tag{10}$$

$$\frac{\partial\phi^{n+1}}{\partial\mathbf{n}}=0 \quad \text{on } \partial\Omega \tag{11}$$

Once ϕ^{n+1} is computed, the divergence-free velocity and the pressure are obtained by:

$$\mathbf{u}^{n+1}=\mathbf{u}_*^{n+1}-\frac{\Delta t}{\rho}\nabla\phi^{n+1} \tag{12}$$

$$p^{n+1}=\phi^{n+1}+p^n-\mu\nabla\cdot\mathbf{u}_*^{n+1} \tag{13}$$

2.1.2. Spatial discretization. The spatial discretization is based on the finite volume method on a velocity–pressure staggered grid of the Marker and Cells type [17]. Pressure unknowns are associated with the cell vertices, whereas velocity components are face centred. A centred scheme of order 2 is used in this study for the inertial and constraint terms.

DOI: 10.1002/fld

The multifrontal sparse direct solver MUMPS [18] is used to solve the linear systems stemming from the velocity prediction and pressure correction steps. BiCGStab(2) coupled with ILUT preconditioner was also successfully tested.

3. AN IMPLICIT METHOD FOR CONNECTING BLOCKS

In order to connect the sub-domain, the missing information is transferred from block to block. Polynomial interpolations are built and integrated as special boundary conditions. The polynomial coefficients of the interpolation are present in the linear system and couple the solution on each block through the interface. The non-conservative interpolation of the variables at the interfaces can be seen as a new implicit boundary condition used for the discretization of the equation at the nodes strictly inside the different blocks.

3.1. Pressure correction step

Two blocks (a) and (b) are considered (see Figure 1). The pressure increment ϕ defined on block (b) is interpolated, which gives the new boundary conditions on block (a). Interpolation is based on the construction of a polynomial basis of a given order. For instance, the interpolation of ϕ at point $M_0(x_0, y_0)$ belonging to block (a) is obtained from the values of ϕ at points $M_i(x_i, y_i)$ on block (b) by the relation:

$$\phi^{(a)}(x_0, y_0) = f_{\text{int}}(\phi^{(b)}) = \sum_{i=1}^{N} F_i(x_0, y_0)\phi^{(b)}(x_i, y_i) \tag{14}$$

The interpolation must now be constructed locally to each node at the interface. The technique consists in building a canonical basis of Q-type of order d, thanks to the neighbourhood of $M_0(x_0, y_0)$. The number of nodes required depends on the order of the chosen polynomial. For instance, for a $Q^{(1)}$ interpolation Figure 1 represents the interpolation of the pressure on node M_0 belonging to the interface of block (a) obtained from the values of pressure on nodes M_1, M_2, M_3 and M_4 belonging to block (b).

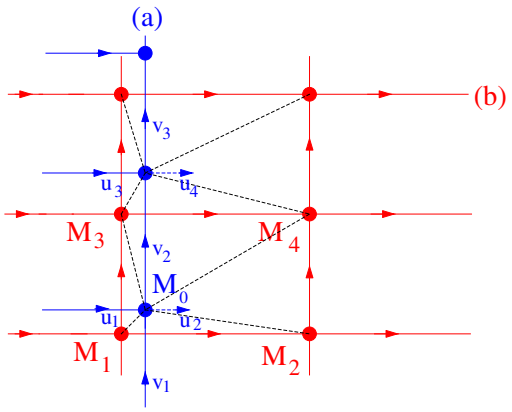


Figure 1. Interpolation of the pressure.

DOI: 10.1002/fld

In order to reduce the values of the coefficients of the polynomial, M_0 is chosen as the centre of the frame. A polynomial $Q_i^{(d)}$ built using the M_i nodes, $1 \leqslant i \leqslant (d+1)^2$ is written:

$$Q_i^{(d)}(x-x_0, y-y_0) = \sum_{m=0}^{d} \sum_{n=0}^{d} a_{mni}(x-x_0)^m (y-y_0)^n \tag{15}$$

$Q_i^{(d)}$ has the following properties:

$$\forall i, j \quad 1 \leqslant i, \quad j \leqslant (d+1)^2, \quad Q_i^{(d)}(x_j - x_0, y_j - y_0) = \delta_{ij} \tag{16}$$

Equation (15) associated with the property (16) can be seen as a line of the $(d+1)^2 \times (d+1)^2$ linear system $A \times B = Id$ with:

$$A = \begin{bmatrix} a_{001} & \cdots & a_{mn1} & \cdots & a_{dd1} \\ \vdots & & \vdots & & \vdots \\ a_{00i} & \cdots & a_{mni} & \cdots & a_{ddi} \\ \vdots & & \vdots & & \vdots \\ a_{00(d+1)^2} & \cdots & a_{mn(d+1)^2} & \cdots & a_{dd(d+1)^2} \end{bmatrix}$$

and

$$B = \begin{bmatrix} (x_1-x_0)^0(y_1-y_0)^0 & \cdots & (x_i-x_0)^0(y_i-y_0)^0 & \cdots & (x_{(d+1)^2}-x_0)^0(y_{(d+1)^2}-y_0)^0 \\ \vdots & & \vdots & & \vdots \\ (x_1-x_0)^m(y_1-y_0)^n & \cdots & (x_i-x_0)^m(y_i-y_0)^n & \cdots & (x_{(d+1)^2}-x_0)^m(y_{(d+1)^2}-y_0)^n \\ \vdots & & \vdots & & \vdots \\ (x_1-x_0)^d(y_1-y_0)^d & \cdots & (x_i-x_0)^d(y_i-y_0)^d & \cdots & (x_{(d+1)^2}-x_0)^d(y_{(d+1)^2}-y_0)^d \end{bmatrix}$$

The inversion of this linear system (one for each node of the interface) is performed during the preparation step of a simulation and provides the values of the matrix A. The value of the pressure ϕ at node $M_0(x_0, y_0)$ reads:

$$\phi^{(a)}(x_0, y_0) = \sum_{i=1}^{(d+1)^2} Q_i^{(d)}(x_0, y_0)\phi^{(b)}(x_i, y_i) \tag{17}$$

The $Q_i^{(d)}$ can be placed in the linear system of the pressure correction step (see Figure 2).

Thus, on a matrix line corresponding to a node at the interface, non-zero elements are the diagonal term and the elements with a column number corresponding to unknowns used to interpolate the pressure.

With this method, the pressure is obviously continuous at the interfaces (up to the order of the polynomial interpolation), but the divergence of the velocity is not null at the interfaces since the velocities u_2, u_4, v_1, v_2 and v_3 (see Figure 1) are not corrected by the pressure correction. Indeed,

DOI: 10.1002/fld

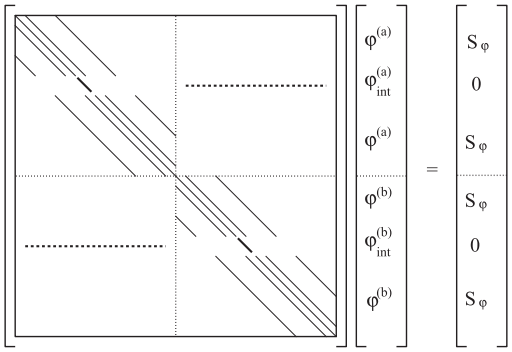


Figure 2. Representation of the pressure correction matrix on 2 blocks.

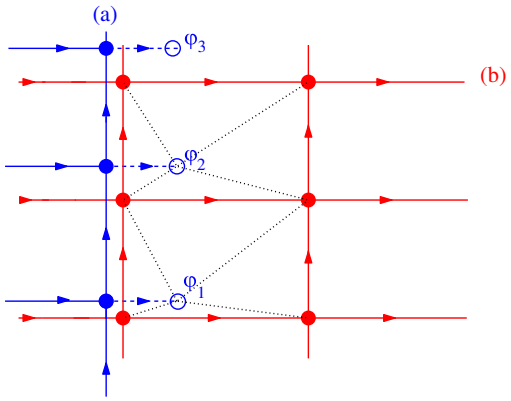


Figure 3. New stencil to ensure incompressibility constraint.

the pressure gradient cannot be well computed on these nodes and introduces an error in Equation (12). In order to circumvent this problem, we propose to increase the overlap between blocks by adding a row of ghost nodes (nodes ϕ_1, ϕ_2 and ϕ_3 in Figure 3). The pressure is now connected from one block to another, thanks to these ghost nodes. The pressure gradient can be computed precisely on velocity nodes at the interfaces. This addition satisfies incompressibility constraint. Nonetheless, the pressure is no longer continuous at the interface. This is probably due to the error in the interpolation of ϕ^{n+1} on the ghost nodes that accumulates on the pressure nodes p^{n+1} at the interface. In order to overcome this problem, a new pressure correction scheme is proposed. The velocity prediction step does not change, but the pressure correction step is modified and a third interpolation step is added. The new scheme reads as follows:

- Velocity prediction step: *Find* $\mathbf{u}_*^{n+1}$ *such that*

$$\rho\left(\frac{\mathbf{u}_*^{n+1}-\mathbf{u}^n}{\Delta t}+\nabla\cdot(\mathbf{u}_*^{n+1}\otimes\mathbf{u}^n)-\mathbf{u}_*^{n+1}\nabla\cdot\mathbf{u}^n\right)=-\nabla p^n+\nabla\cdot\mu(\nabla\mathbf{u}_*^{n+1}+\nabla^t\mathbf{u}_*^{n+1}) \quad \text{in } \Omega \tag{18}$$

$$\mathbf{u}_*^{n+1}=\mathbf{0} \quad \text{on } \partial\Omega \tag{19}$$

DOI: 10.1002/fld

- Pressure correction step: *Find* $\mathbf{u}^{n+1}$ and ϕ^{n+1} such that

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}_*^{n+1}}{\Delta t}+\nabla\phi^{n+1}=\mathbf{0} \quad \text{in } \Omega \tag{20}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0 \quad \text{in } \Omega \tag{21}$$

$$\mathbf{u}^{n+1}\cdot\mathbf{n}=0 \quad \text{on } \partial\Omega \tag{22}$$

  with:

$$\phi^{n+1}=\tilde{p}^{n+1}-p^n+\mu\nabla\cdot\mathbf{u}_*^{n+1} \tag{23}$$

- Interpolation step: *Compute* p^{n+1} *such that*

$$p^{n+1}=\tilde{p}^{n+1} \quad \text{in } \Omega/\Gamma \tag{24}$$

$$p^{n+1}=f_{\text{int}}(\tilde{p}^{n+1}) \quad \text{on } \Gamma \tag{25}$$

  where Γ is the whole interface between blocks and f_{int} represents the interpolation function on Γ.

This new scheme ensures continuity of the pressure through the interfaces and still respects the incompressibility constraint.

3.2. *Velocity prediction step*

Interpolation of the velocity is more difficult since it is a vector field. If the blocks do not have the same orientation, both velocity components are needed to compute a single component of the velocity field on the interface. For a precise description of the method, particularly the interpolation technique on cartesian blocks with any orientation or on curvilinear blocks, the reader is referred to [14]. Previously, the interpolation of the normal component of the velocity field at the interface was performed on pressure nodes, whereas the tangential component was interpolated at the velocity nodes. In the present case, both components are interpolated at the velocity nodes (see Figure 4 in case of a $Q^{(1)}$ interpolation).

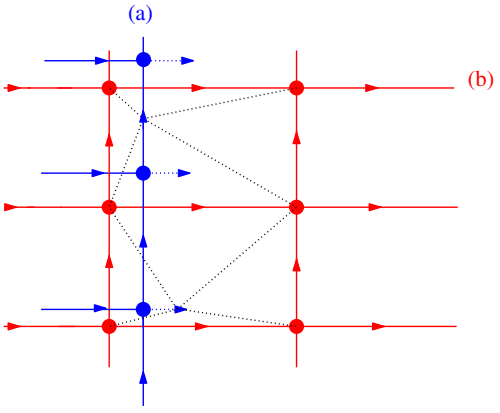


Figure 4. New interpolation of the velocity vector on a staggered grid.

DOI: 10.1002/fld

In the next section, many computations have been carried out to assess the proposed method and to exhibit its improvements in comparison with [14]. $Q^{(2)}$ polynomial interpolation has been systematically used. This leads to a 5-point stencil for pressure and to a 24-point stencil for velocity.

4. NUMERICAL RESULTS

In this section, a series of numerical experiments was performed to evaluate the accuracy and the efficiency of the method. The test cases considered include steady and unsteady flows. First, we consider a Poiseuille flow and the Green–Taylor vortex example to verify the spatial and time order of convergence of the method. Then, the lid-driven cavity is studied to assess the improvement of the method in comparison with [14]. Finally, flows past a two-dimensional obstacle and past a triangular cylinder are considered.

4.1. Poiseuille flow

Simulations were carried out in several non-conforming block-structured meshes represented in Figure 5. The Reynolds number is $Re = 100$. At the inlet, we impose a parabolic profile for the axial velocity u and zero for the velocity v, whereas at the outlet, a Neumann condition is imposed on both velocity components. Since the analytical solution is order 2, the errors are close to computer accuracy if a $Q^{(2)}$ interpolation is used.

4.2. Green–Taylor vortex

The main point of interest with this flow is that, unlike in Poiseuille flow, the inertial term is not null. The Green–Taylor vortex is modified to obtain a stationary solution not identically null [19]. In [14], the authors could not validate the pressure because the divergence of the velocity was not null and accumulated in the pressure. We will see that the new scheme proposed clearly improves these inconveniences.

The momentum equations are enriched by the following source term:

$$\mathbf{S} = \begin{cases} -\dfrac{\pi^2 \mu}{2H^2} \cos\left(\dfrac{\pi x}{2H}\right) \sin\left(\dfrac{\pi y}{2H}\right) \\ -\dfrac{\pi^2 \mu}{2H^2} \sin\left(\dfrac{\pi x}{2H}\right) \cos\left(\dfrac{\pi y}{2H}\right) \end{cases}$$

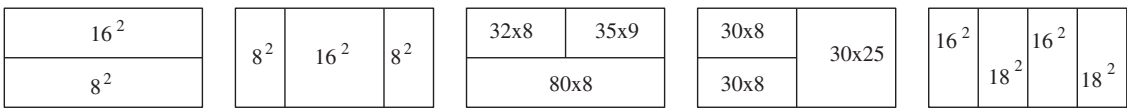


Figure 5. Representation of the different blocks for Poiseuille flow study (number of elements is indicated).

DOI: 10.1002/fld

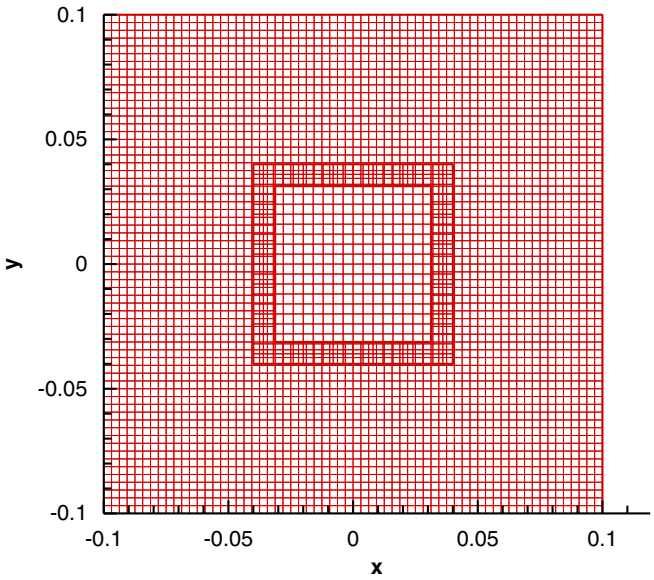


Figure 6. Block-structured mesh for the Green–Taylor flow study.

The solution is:

$$u(x,y,t) = -\cos\left(\frac{\pi x}{2H}\right)\sin\left(\frac{\pi y}{2H}\right)(1-\mathrm{e}^{-\pi^2 \nu t/2H^2})$$

$$v(x,y,t) = +\sin\left(\frac{\pi x}{2H}\right)\cos\left(\frac{\pi y}{2H}\right)(1-\mathrm{e}^{-\pi^2 \nu t/2H^2})$$

$$p(x,y,t) = -\frac{\rho}{2}\left(\cos^2\left(\frac{\pi x}{2H}\right)+\cos^2\left(\frac{\pi y}{2H}\right)\right)(1-2\mathrm{e}^{-\pi^2 \nu t/2H^2}+\mathrm{e}^{-\pi^2 \nu t/H^2})$$

The boundary conditions are obtained directly from the analytical solution and are modified at each time step. The test case was run with two non-conforming structured blocks (see Figure 6).

A time convergence study was carried out. Figure 7 represents the errors at time $T=1\,\mathrm{s}$ as a function of the time step Δt. We observe a slope in log/log scale equal to 1 that is coherent with the order of the scheme used (Euler time scheme and linearization at order 1).

4.3. Lid-driven cavity

The lid-driven cavity problem has long been used as a test or validation case for new codes or new methods. The standard case is fluid contained in a square domain with three wall sides and one moving side (with velocity tangential to the side). In the present paper, we refer to the works of Botella and Peyret [20]. For this computation, a Reynolds number of 1000 is used.

The mesh is divided into five non-conforming structured blocks: the first block occupies the main part of the domain; the lower (upper) blocks have step spaces a third (half) the size of the main block (see Figure 8). The mesh is refined in the corners to have a better description of the flow. The refinement is clearly a strong point of the multiblock strategy. The number of elements is 121 344.

DOI: 10.1002/fld

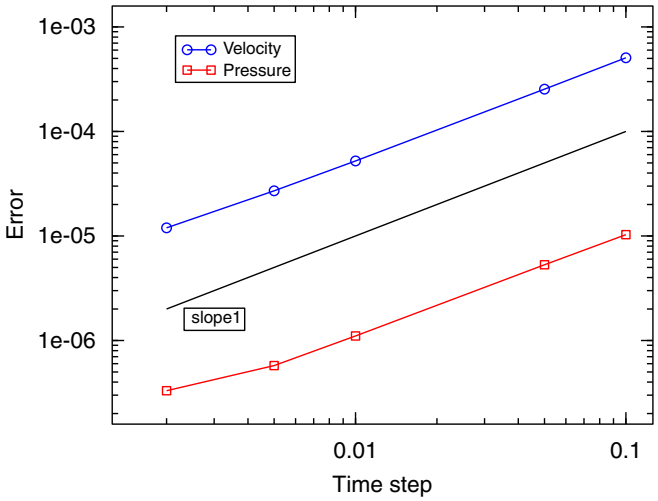


Figure 7. $L^2(\Omega)$ error norm as a function of the time step for the Green–Taylor flow study.

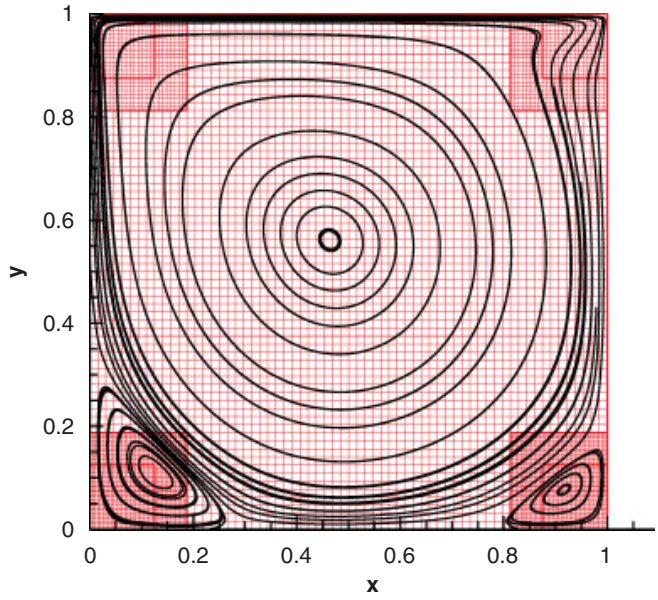


Figure 8. Streamlines on an block-structured mesh for the lid-driven cavity.

The main results found in the literature are given on the intensity and position of the vortices. The results are obtained with convergence criteria on stationarity below 10^{-12}. Our results are compared with the results obtained in [14] to exhibit the improvements of the method.

4.3.1. Position and intensity of the vortex. Tables I–III that represent the position of the vortices and the intensity of the streamlines show the good behaviour of the method in comparison with the results of the literature. We can observe in the two first lines of the table that both methods

DOI: 10.1002/fld

Table I. Intensity and position (x, y) of the main vortex.

Reference	Mesh size	Maximum streamline	x	y
Present	$121\,344 \simeq 348^2$	0.11885	0.4687	0.5664
Romé *et al.* [14]	$121\,344 \simeq 348^2$	0.11877	0.4687	0.5664
Botella and Peyret [20]	128^2	0.11894	0.4692	0.5652

Table II. Intensity and position (x, y) of the left secondary vortex.

Reference	Mesh size	Minimum streamline	x	y
Present	$121\,344 \simeq 348^2$	-1.7285×10^{-3}	0.1354	0.1119
Romé *et al.* [14]	$121\,344 \simeq 348^2$	-1.7285×10^{-3}	0.1354	0.1120
Botella and Peyret [20]	128^2	-1.7297×10^{-3}	0.1360	0.1118

Table III. Intensity and position (x, y) of the left ternary vortex.

Reference	Mesh size	Maximum streamline	x	y
Present	$121\,344 \simeq 348^2$	4.6742×10^{-8}	0.00757	0.00755
Romé *et al.* [14]	$121\,344 \simeq 348^2$	4.7314×10^{-8}	0.00781	0.00781
Botella and Peyret [20]	128^2	5.0399×10^{-8}	0.00768	0.00765

give very close results. The utility of the refinement is clear since it provides an amplitude of 10^{-8} for the right ternary vortex.

4.3.2. Velocity and pressure profiles. In Figure 9, the velocity profiles present a good accordance with those obtained by Botella and Peyret [20] and from a monoblock mesh with a Chebyshev polynomial step size variation. In [14], the authors observed that the pressure was discontinuous on the interface of the upper blocks because the incompressibility constraint did not equal zero. Figure 10 clearly shows the improvement of the method since the pressure is now continuous.

4.4. Laminar flow over a two-dimensional obstacle

The experimental configuration is presented in Figure 11. This flow was experimentally studied by Carvalho *et al.* [21]. The Reynolds number based on the height of the obstacle and mean axial velocity is 82.5. A recirculation appears behind the obstacle and when the Reynolds number increases, a second recirculation appears on the upper wall (see Figure 12). The boundary conditions at the inlet are prescribed as a parabolic profile for the axial velocity u and zero for the velocity v. At the outlet, a Neumann condition is imposed on both velocity components. In the present study, a mesh composed of three structured blocks is considered. Even if this configuration can be treated with an uniformly refined grid, we have chosen this test case to prove the robustness of our approach by putting the interface between blocks in a strong shear zone. Indeed, for complex flows and geometries, the fact that vortices cross the interface is very frequent. The grid is fine around the obstacle and stretched near the inlet and the outlet (see Figure 12). The total number of elements is 131 040.

DOI: 10.1002/fld

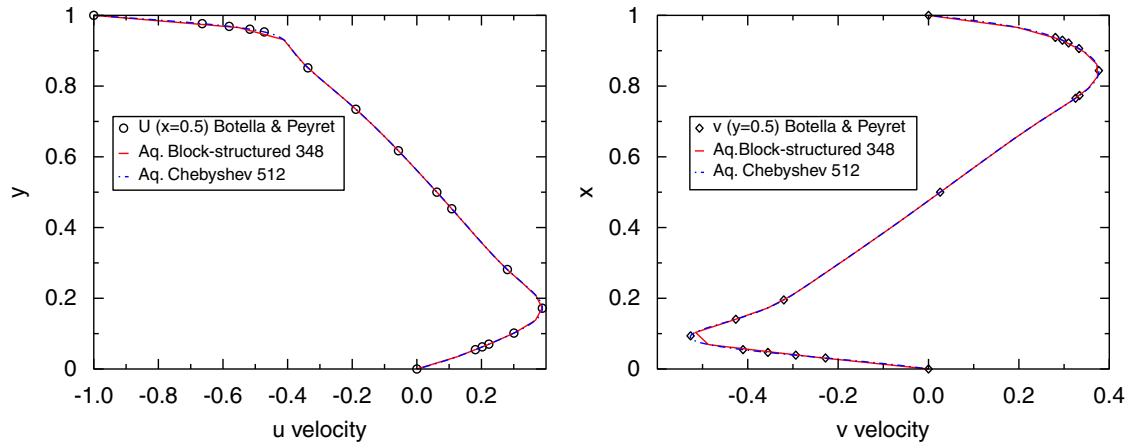


Figure 9. u profiles at $x=0.5$ (left) and v profiles at $y=0.5$ (right).

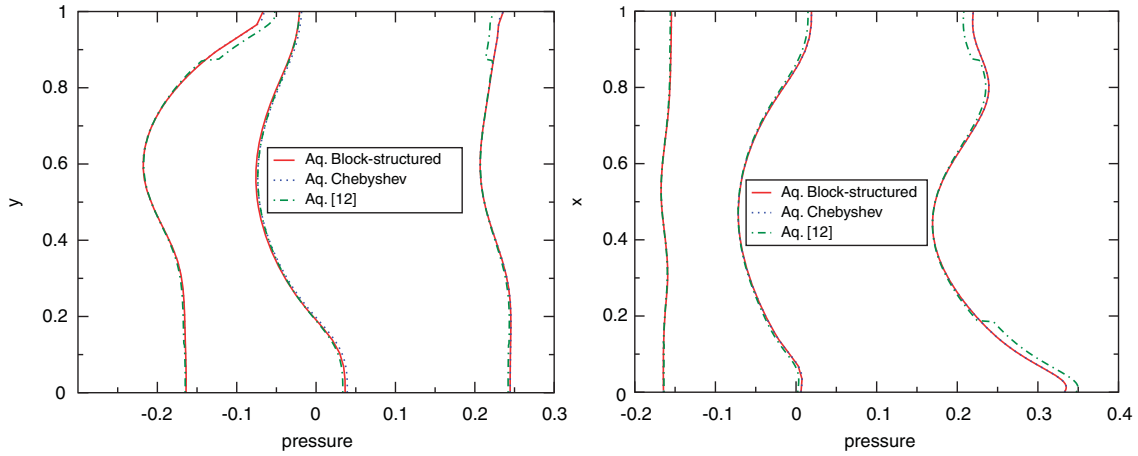


Figure 10. Pressure profiles at $x=0.1, 0.5, 0.9$ (left) and at $y=0.1, 0.5, 0.9$ (right).

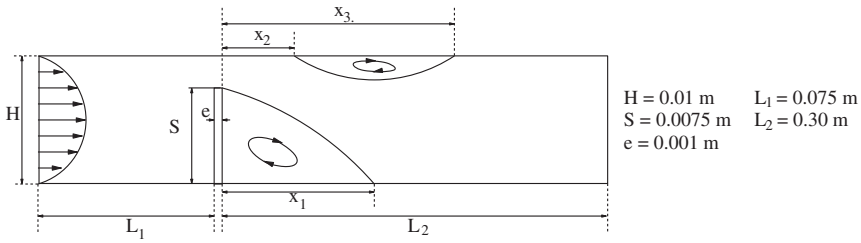


Figure 11. Global features of the flow over a two-dimensional obstacle.

Figure 13 presents the axial velocity u at different locations x/S measured [21] and computed. Table IV compares the reattachment lengths. Satisfactory results can be observed in comparison with the experimental data except for the reattachment length x_2/S. Several authors [22–24] have

DOI: 10.1002/fld

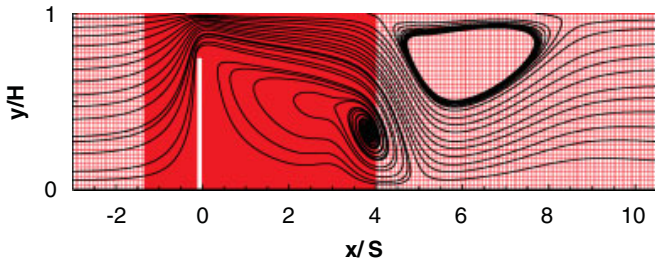


Figure 12. Streamlines on a block-structured mesh for the flow over a two-dimensional obstacle.

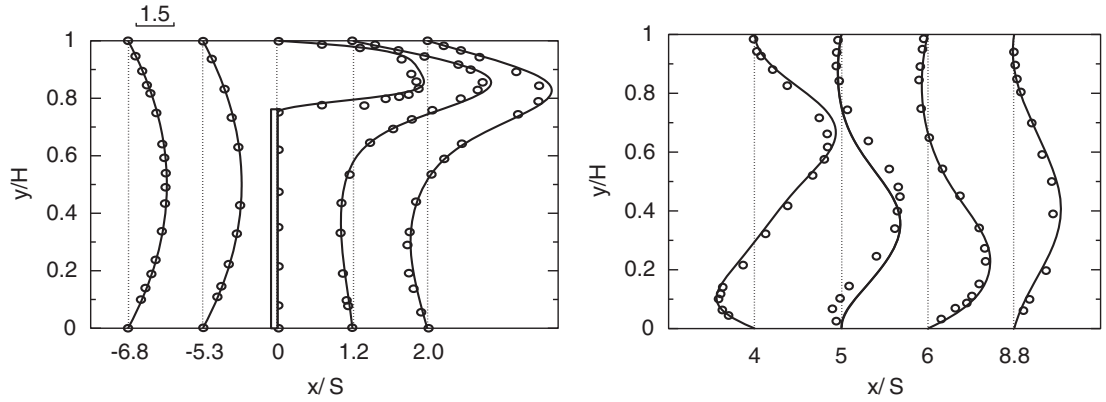


Figure 13. Comparison between calculation (—) and experimental data [21] (○) for flow over the fence ($Re = 82.5$).

Table IV. Comparison of the reattachment lengths.

	x_1/S	x_2/S	x_3/S
Present	4.96	4.12	8.38
Aq. monoblock 1800×200	4.99	4.01	9.19
Carvalho *et al.* [21]	5.00	2.01	10.4

considered this flow to validate their scheme but they have only compared the axial velocity u at different locations x/S and none of them has measured the recirculation lengths. Consequently, we cannot really verify if the difference on x_2/S comes from our computation or from the measurement in [21]. At least, multiblock results are coherent (see Table IV) with those obtained on a fine monoblock grid, solved with a different velocity–pressure coupling (augmented lagrangian).

4.5. Flow past a triangular cylinder

Flow past a equilateral triangular cylinder was studied. The Reynolds number is based on the side of the triangular cylinder ($h = 1$) and the axial velocity inlet ($u = 0.5$). We focussed on two flow

DOI: 10.1002/fld

ranges according to the value of the Reynolds number in relation to its critical value Re_c:

- $Re < Re_c$: the flow is stationary. Two steady symmetrical vortices can be observed behind the cylinder: their size increases with increasing Re.
- $Re \geqslant Re_c$: the flow becomes unsteady and periodic. Two vortices form at the rear-end vertices of the cylinder and are shed alternately.

Jackson [25] studied the onset of vortex shedding in flow past variously shaped bodies. For an isosceles triangle with base 1 and height 0.8, he reported a critical Reynolds as 34.318 and a corresponding Strouhal number as 0.13554. Zielinska and Wesfreid [26] computed flow past an equilateral triangle with a blockage ratio equal to $\frac{1}{15}$ and found a critical Reynolds number of 38.3. De and Dalal [27] carried out a similar study and calculated a critical Reynolds number of 39.9 for a blockage ratio of $\frac{1}{20}$. This case was chosen here because most studies have dealt with flow past circular or square cylinders and laminar flow past a triangular cylinder has not been intensively studied so far. Moreover, this configuration is well adapted to validate and illustrate the utility of block-structured meshes.

4.5.1. Parameters of the case test. Figure 14 represents the block-structured mesh used in this case. The grid is fine around the cylinder and space step size increases in front of and behind it. The number of grid nodes distributed over a side of the cylinder is 100. The total number of elements is 137 980. At the inlet, a flat profile is imposed for the axial velocity u and zero for the velocity v. At the outlet, a Neumann condition is imposed on both velocity components. We will compare the results with [27].

4.5.2. Steady flow. The streamlines in the vicinity of the cylinder for two Reynolds numbers are shown in Figure 15.

To assess the method, the recirculation length (L_r) defined by the reattachment of the fluid was measured and a linear relationship between L_r and Re obtained in [27] (see Figure 16). The results seem to be in good accordance with those presented in [27].

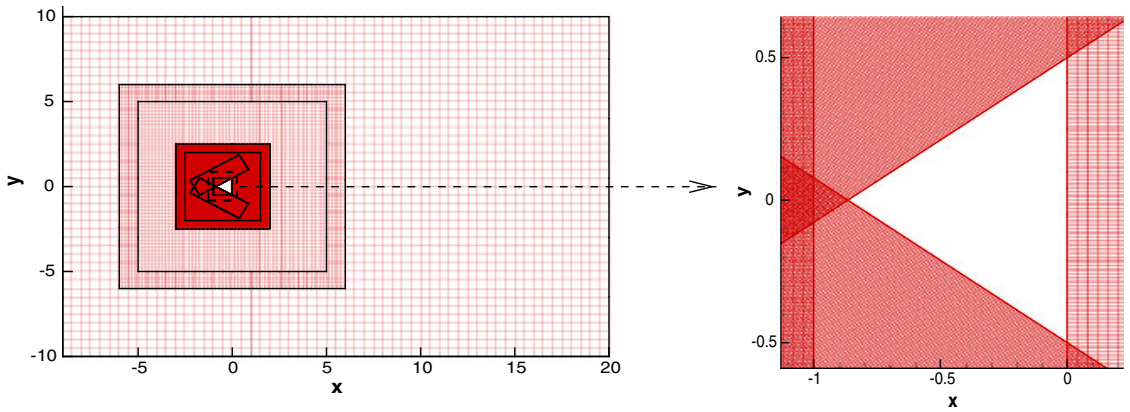


Figure 14. Block-structured mesh for flow past a triangular cylinder.

DOI: 10.1002/fld

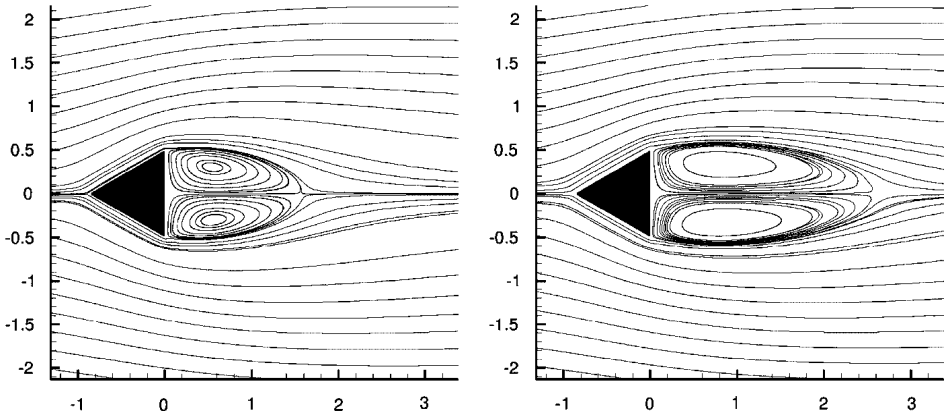


Figure 15. Steady-state streamlines $Re=20$ (left), $Re=35$ (right).

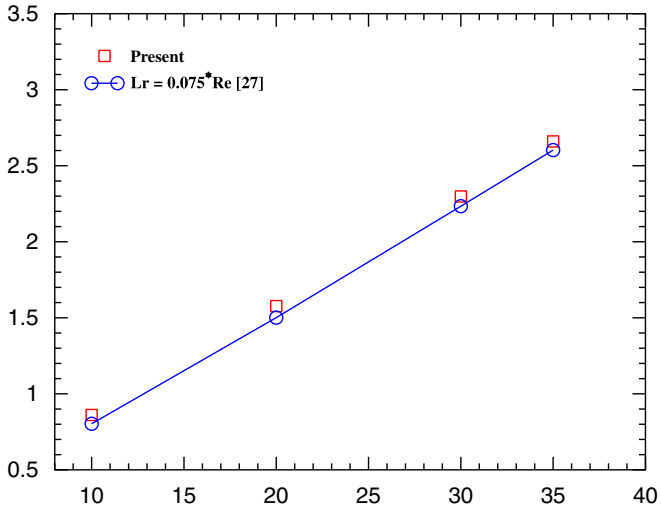


Figure 16. $L_r - Re$ relationship.

Table V. Comparison of the results for $Re=100$.

	C_D	C_{D_p}	$C_{L_{rms}}$	St
Present	1.6698	1.3579	0.2626	0.1960
De and Dalal [27]	1.7549	1.2986	0.2974	0.1962

4.5.3. Unsteady and periodic flow. The flow becomes unsteady and periodic for $Re \geqslant 40$. For $Re=100$, the time-average drag coefficient (C_D), time-average pressure drag coefficient (C_{D_p}), rms of the lift coefficient ($C_{L_{rms}}$) and the Strouhal number (St) are compared with [27]. We can see in Table V difference below 5% except for the $C_{L_{rms}}$ where the gap is 11%. It can be explained

DOI: 10.1002/fld

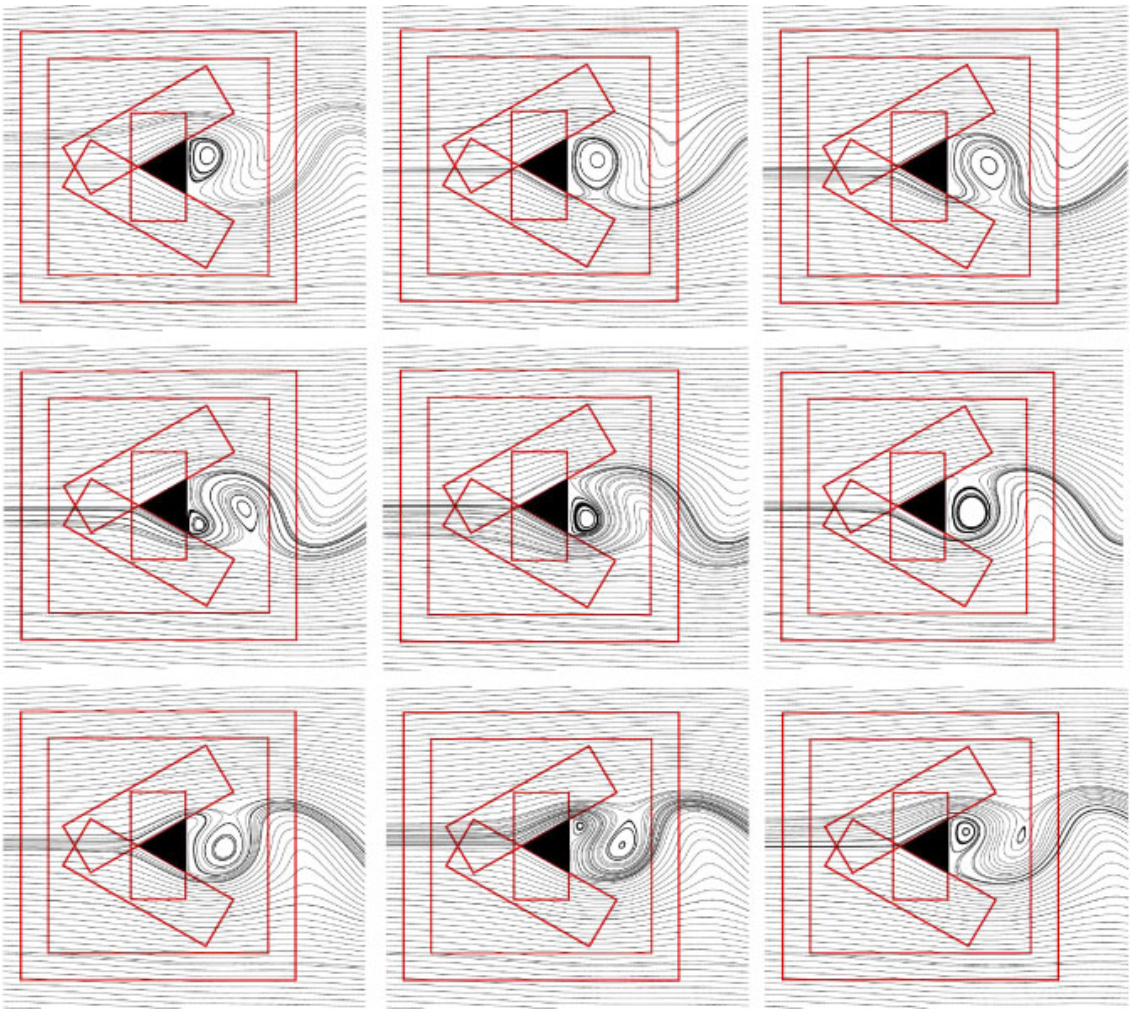


Figure 17. Streamlines for the flow past a triangular cylinder during a period at $Re = 100$ (it reads from left to right and vertically).

by the difference of the mesh size and the numerical method used by the authors. We can notice that ensuring the continuity of the pressure between blocks allowed us to compute the drag and lift coefficients even if the contours of the geometry are described by several blocks (contrary to the previous method where jumps of pressure between blocks were observed).

Finally, Figure 17 shows the streamlines around the triangular cylinder during a period at $Re = 100$ giving prominence to the periodicity of the flow.

5. CONCLUSION

In this paper, we have proposed an improvement of a method first introduced by Romé *et al.* in [14]. Both methods deal with a domain decomposition technique for non-conforming and

DOI: 10.1002/fld

overlapping block-structured meshes. They are non-iterative and based on a implicit non-conservative interpolation of the variables at the interfaces. The linear systems are modified in comparison with those obtained on a monoblock mesh since lines are added to take into account the connectivity between blocks.

The main difference between the two methods is the velocity–pressure coupling. In [14], the authors dealt with this coupling by an augmented Lagrangian method. In some cases, the divergence of the velocity at the interfaces between blocks was not null leading to a discontinuity for the pressure. In the present study, velocity and pressure were coupled by a pressure correction scheme. The divergence of velocity is now null over the whole domain and the discontinuity of pressure has disappeared. In many cases, the gradient of the pressure is sufficient to carry on computations but an accurate pressure is needed to compute physical parameters such as drag and lift coefficients for example. Consequently, the proposal method improves this point. Several numerical tests were carried out in order to validate the method. They clearly showed its feasibility and accuracy.

ACKNOWLEDGEMENTS

The authors thank Professor Mejdi Azaïez for his critical reading, helpful suggestions and fruitful discussions that guided us to achieve this work. We acknowledge calculation facilities financially supported by the Conseil Régional d'Aquitaine and the French Ministry of Research and Technology.

REFERENCES

1. Quarteroni A, Valli A. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press: Oxford, 1999.
2. Schwarz HA. Über einen Grenzbergang durch alternirendes Verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich* 1870; **15**:272–286.
3. Lions PL. On the Schwarz alternating method III: a variant for nonoverlapping subdomains. *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1990.
4. Bernardi C, Maday Y, Patera AT. Domain decomposition by the mortar element method. In *Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters*, Kaper HG, Garbey M (eds). NATO ASI Series C, vol. 384. Kluwer Academic Publishers: Dordrecht, 1993; 269–286.
5. Cai XC, Dryja M, Sarkis M. Overlapping nonmatching grid mortar element methods for elliptic problems. *SIAM Journal on Numerical Analysis* 1999; **36**:581–606.
6. Benek JA, Buning PG, Steger JL. A 3-D chimera grid embedding technique. *AIAA 85-1523CP*, 1985.
7. Steger JL, Benek JA. On the use of composite grid schemes in computational aerodynamics. *Computer Methods in Applied Mechanics and Engineering* 1987; **64**:301–320.
8. Lilek Z, Muzaferija S, Peric M, Seidl V. An implicit finite-volume method using nonmatching blocks of structured grid. *Numerical Heat Transfer* 1997; **32**:385–401.
9. Usera G, Vernet A, Ferré JA. A parallel block-structured finite volume method for flows in complex geometry with sliding interfaces. *Flow Turbulence Combustion* 2008; **81**:471–495.
10. Tang HS, Casy Jones S, Sotiropoulos F. An overset-grid method for 3D unsteady incompressible flows. *Journal of Computational Physics* 2003; **191**:567–600.
11. Tang HS. Study on a grid algorithm for solution of incompressible Navier–Stokes equations. *Computers and Fluids* 2006; **35**:1372–1383.
12. Henshaw WD, Watson TJ. A fourth-order accurate method for the incompressible Navier Stokes equations on overlapping grids. *Journal of Computational Physics* 1994; **113**:13–25.
13. Romé C, Glockner S. An implicit multiblock coupling for the incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 2005; **47**:1261–1267.
14. Romé C, Glockner S, Caltagirone JP. Resolution of the Navier–Stokes equations on block-structured meshes. *International Journal for Numerical Methods in Fluids* 2007; **54**:1239–1268.

DOI: 10.1002/fld

15. Fortin M, Glowinski R. *Méthodes de Lagrangien Augmenté*, *Applicationà la résolution numérique de problèmes aux limites*. Collection Méthodes Mathématiques de l'Informatique. Dunod: Paris, 1982.
16. Timmermans LJP, Minev PD, Van De Vosse FN. An approximate projection scheme for incompressible flow using spectral elements. *International Journal for Numerical Methods in Fluids* 1996; **22**:673–688.
17. Harlow FH, Welch JE. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids* 1965; **8**:2182–2189.
18. Amestoy PR, Duff IS, L'Excellent JY. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering* 2000; **184**:501–520.
19. Caltagirone JP, Breil J. Sur une méthode de projection vectorielle pour la résolution des équations de Navier–Stokes. *Comptes Rendus de l'Académie des Sciences*, *Série II*, *Paris* 1999; **327**:1179–1184.
20. Botella O, Peyret R. Benchmark spectral results on the lid-driven cavity flow. *Computers and Fluids* 1998; **4**:421–433.
21. Carvalho MG, Durst F, Pereira JC. Predictions and measurements of laminar flow over two-dimensional obstacles. *Applied Mathematical Modelling* 1987; **11**:23–34.
22. Song B, Liu GR, Lam KY, Amano RS. On a higher-order bounded discretization scheme. *International Journal for Numerical Methods in Fluids* 2000; **32**:881–897.
23. N'Dri D, Garon A, Fortin A. Incompressible Navier–Stokes computations with stable and stabilized space–time formulations: a comparative study. *Communications in Numerical Methods in Engineering* 2002; **18**:495–512.
24. Rida S, McKenty F, Meng FL, Reggio M. A staggered control volume scheme for unstructured triangular grids. *International Journal for Numerical Methods in Fluids* 1997; **25**:697–717.
25. Jackson CP. A finite-element study of the onset of vortex shedding in flow past variously shaped bodies. *Journal of Fluid Mechanics* 1987; **182**:23–45.
26. Zielinska BJ, Wesfreid JE. On the spatial structure of global modes in wake flow. *Physics of Fluids* 1995; **7**:1418–1424.
27. De AK, Dalal A. Numerical simulation of unconfined flow past a triangular cylinder. *International Journal for Numerical Methods in Fluids* 2006; **52**:801–821.

DOI: 10.1002/fld

4 Reduction of the discretization stencil of direct forcing immersed boundary methods on rectangular cells : The ghost node shifting method

Journal of Computational Physics 364 (2018) 18–48

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Reduction of the discretization stencil of direct forcing immersed boundary methods on rectangular cells: The ghost node shifting method

Joris Picot [a,*], Stéphane Glockner [b]

[a] Université de Bordeaux, I2M, UMR 5295, F-33400 Talence, France
[b] Bordeaux INP, I2M, UMR 5295, F-33400 Talence, France

ARTICLE INFO

Article history:
Received 1 September 2017
Received in revised form 15 February 2018
Accepted 23 February 2018
Available online xxxx

Keywords:
Immersed boundary method
Direct forcing
Discretization stencil
Poisson problem
Incompressible Navier–Stokes
Boundary conditions

ABSTRACT

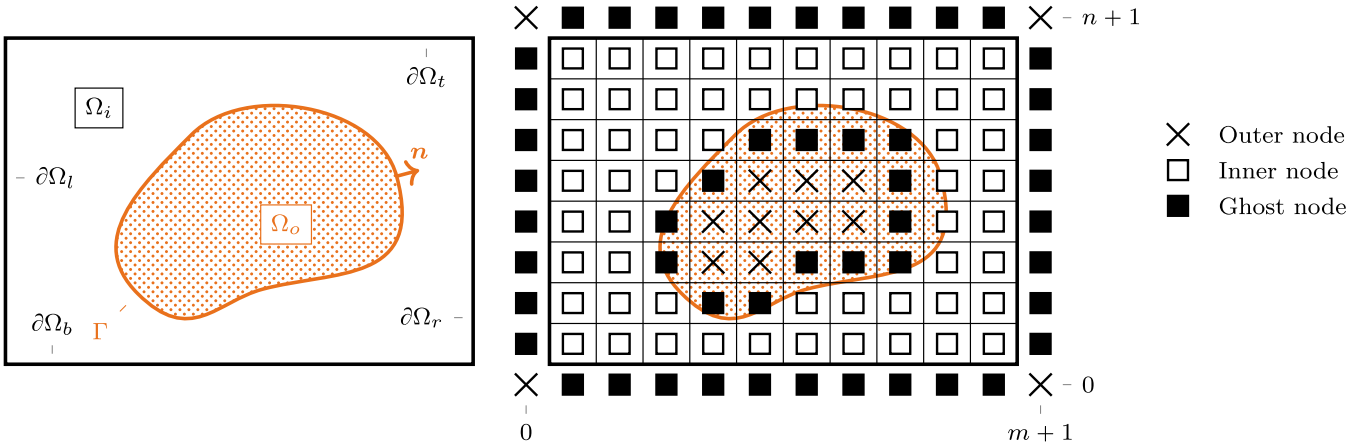
We present an analytical study of discretization stencils for the Poisson problem and the incompressible Navier–Stokes problem when used with some direct forcing immersed boundary methods. This study uses, but is not limited to, second-order discretization and Ghost-Cell Finite-Difference methods. We show that the stencil size increases with the aspect ratio of rectangular cells, which is undesirable as it breaks assumptions of some linear system solvers. To circumvent this drawback, a modification of the Ghost-Cell Finite-Difference methods is proposed to reduce the size of the discretization stencil to the one observed for square cells, i.e. with an aspect ratio equal to one. Numerical results validate this proposed method in terms of accuracy and convergence, for the Poisson problem and both Dirichlet and Neumann boundary conditions. An improvement on error levels is also observed. In addition, we show that the application of the chosen Ghost-Cell Finite-Difference methods to the Navier–Stokes problem, discretized by a pressure-correction method, requires an additional interpolation step. This extra step is implemented and validated through well known test cases of the Navier–Stokes equations.

1. Introduction

The immersed boundary method has been proven a successful treatment of complex boundaries in numerical simulations. The difficult generation of body-fitted grids is replaced by a modification of the governing equations near the boundaries, thus benefits of structured grids can be exploited. In addition, the immersed boundary method is particularly attractive when dealing with moving boundaries as it avoids re-meshing. Many different variants have been developed since the original immersed boundary method [1] as they explored coupling with different physical problems, different regimes, etc. Now the best approach to choose depends on physics constraints one wants to model. The review of Mittal and Iaccarino [2] explains this in details. The Direct Forcing approach, adopted in this article, first discretizes governing equations into a linear system; then applies boundary conditions near immersed boundaries to complete the linear system. This approach is similar to ordinary boundary conditions, but the non-Cartesian immersed boundaries leads to more complex interpolations. Both finite-difference discretization and finite-volume discretization can be used to apply immersed boundary conditions. This article focuses on specific aspects of the Ghost-Cell Finite-Differences approach, in the versions

* Corresponding author.
E-mail addresses: joris.picot@ens-lyon.fr (J. Picot), glockner@bordeaux-inp.fr (S. Glockner).

https://doi.org/10.1016/j.jcp.2018.02.047
0021-9991/



(a) Example of computational domain Ω split in an inner part Ω_i and an outer part Ω_o by the immersed boundary Γ.

(b) Example of Cartesian grid of resolution $m \times n$ with domain boundary extensions build upon Example Fig. 1a. Different symbols are used to distinguish inner, ghost, and outer cell nodes.

Fig. 1. Definitions and notations of the computational domain, the immersed boundary, the mesh, and node types.

proposed by Mittal et al. [3] and Coco and Russo [4], and applies them to the Poisson and Navier–Stokes problem, though this approach can be used to any boundary value problem.

Some linear system solvers constrain the matrix profile, i.e. cells allowed to be nonzero, to take advantage of properties that allow to achieve the highest performance in large parallel computations. For instance, geometric multi-grid algorithms SMG [5,6] and PFMG [7,8] of the *hypre* library can be used only with band matrices with 9/27 points stencils in 2D/3D for SMG and PFMG. More generic solvers and preconditioners such as the algebraic BoomerAMG of the *hypre* library are less efficient (see [9–11]) even if they are competitive. In addition, band matrices requires less memory than more generic sparse matrix format, such as Compressed Row Storage, because their topology do not need to be stored for each line of the matrix. The Ghost-Cell Finite-Differences methods proposed by Mittal et al. [3] and Coco and Russo [4] do not fit in band matrices generated by the discretization of the equations, whose stencil is broken for rectangular cells. These methods are based on the "closest point of the boundary" (as explained in Sec. 2.2) which does not take into account the properties of the numerical grid. This article proposes a modification of these methods to adapt them on band matrices in the case of rectangular cells. This article is organized as follows: the two Ghost-Cell Finite-Differences methods [3,4] are presented in Sect. 2 through the Poisson problem. Their impact on the discretization stencil is studied in Sect. 3, and the Ghost Node Shifting Method is defined in Sect. 4. Numerical simulations validate the proposed method in Sect. 5 for the Poisson problem with Dirichlet and Neumann boundary conditions. Finally, the Ghost-Cell Finite-Differences methods are introduced into the incompressible Navier–Stokes equations in Sect. 6 and some validations are presented in Sect. 7.

2. Immersed boundary methods with the Poisson problem

This section presents the Finite-Differences Ghost-Fluid immersed boundary methods applied to the Poisson problem, as found in the literature. For the sake of simplicity, methods are described in the 2-dimensional real space.

2.1. Problem description and discretization

The *computational domain* Ω is a rectangle of the 2-dimensional real space. It is split into two sub-domains by the *immersed boundary* Γ: one is the *inner part* Ω_i, while the other is the *outer part* Ω_o. The boundary Γ is oriented by a unit vector $\boldsymbol{n}$ from the outer domain to the inner domain. The *domain boundary* $\partial\Omega$ decomposes in four line segments $\partial\Omega_b$, $\partial\Omega_r$, $\partial\Omega_t$, $\partial\Omega_l$, where the letters b, r, t, l stand for bottom, right, top, left, respectively. These definitions are illustrated in Fig. 1a.

Poisson problem. We consider $u : \Omega_i \to \mathbb{R}$, a scalar field restricted to the inner domain, solution of the Poisson equation with a known *source field* $f : \Omega_i \to \mathbb{R}$:

$$\Delta u = f\,, \qquad \text{in } \Omega_i\,, \tag{1a}$$

$$u = D\,, \qquad \text{on } \partial\Omega_D \cup \Gamma_D\,, \tag{1b}$$

$$\partial_{\boldsymbol{n}} u = N\,, \qquad \text{on } \partial\Omega_N \cup \Gamma_N\,. \tag{1c}$$

Here boundaries have been split into a "Dirichlet" part $\partial\Omega_D \cup \Gamma_D$ and a "Neumann" part $\partial\Omega_N \cup \Gamma_N$; boundary conditions $D : \Gamma \to \mathbb{R}$ and $N : \Gamma \to \mathbb{R}$ are known.

Numerical discretization. The computational domain is partitioned into *cells* with a Cartesian arrangement of size $m \times n$. The cell boundaries do not conform with Γ. The partitioning is *uniform,* i.e. the *cell sizes* h_x, h_y do not depend on the cell index; however, the cells can be either *square*: $h_x = h_y$, or *rectangular*: $h_x \neq h_y$. A row of cells is added across each domain boundary to handle the domain boundary conditions; hence the total grid size is $(m+2) \times (n+2)$. Finally, each cell of index (i, j) is associated to a *cell node* $\boldsymbol{X}_{i,j}$, with $0 \leq i \leq m+1$ and $0 \leq j \leq n+1$. Fig. 1b shows the discretization of Ω.

We assign types to nodes to know on which side of the immersed boundary the node is, and to apply different equations on them. The *cell type* of the cell node $\boldsymbol{X}_{i,j}$ is

1. if $\boldsymbol{X}_{i,j} \in \Omega_i$, then $\boldsymbol{X}_{i,j}$ is an *inner node;*
2. else, if $\{\boldsymbol{X}_{i-1,j}, \boldsymbol{X}_{i+1,j}, \boldsymbol{X}_{i,j-1}, \boldsymbol{X}_{i,j+1}\} \cap \Omega_i \neq \emptyset$, then $\boldsymbol{X}_{i,j}$ is a *ghost node;* (2)
3. else, $\boldsymbol{X}_{i,j}$ is an *outer node.*

Fig. 1b also illustrates how node types are defined. As explained below, the ghost node in Eq. (2) represents nodes that will need a special treatment to close the linear system. By extension: if $\boldsymbol{X}_{i,j}$ is an inner node (respectively, a ghost node, an outer node), then the cell at (i, j) is called an *inner cell* (respectively, a *ghost cell*, an *outer cell*). Finally, we define the *set of inner cell indices* C_i, the *set of ghost cell indices* C_g, and the *set of outer cell indices* C_o, accordingly.

Equation (1a) discretizes at each inner node using standard, second-order finite differences

$$\frac{U_{i-1,j} - U_{i,j}}{h_x^2} + \frac{U_{i+1,j} - U_{i,j}}{h_x^2} + \frac{U_{i,j-1} - U_{i,j}}{h_y^2} + \frac{U_{i,j+1} - U_{i,j}}{h_y^2} = F_{i,j} + \mathcal{O}(h^2)\,, \qquad \forall (i,j) \in C_i\,, \tag{3}$$

where $h = \max\{h_x, h_y\}$; F and U are *discrete fields* of size $(m+2) \times (n+2)$ such as $F_{i,j} = f(\boldsymbol{X}_{i,j})$ and $U_{i,j} = u(\boldsymbol{X}_{i,j})$ at inner cells. Note that we choose $U_{i,j}$ to be the field value of the *exact* solution at $\boldsymbol{X}_{i,j}$ of the continuous equation Eq. (1); the truncation term $\mathcal{O}(h^2)$ in Eq. (3) accounts for the difference between finite differences and exact derivatives. We also write Eq. (3) in the matrix form $(LU)_{i,j} = F_{i,j} + \mathcal{O}(h^2), \forall (i,j) \in C_i$. Equation (3) is not closed as there are some $U_{i-1,j}$, $U_{i+1,j}$, $U_{i,j-1}$, or $U_{i,j+1}$ that do not correspond to inner nodes; these nodes correspond exactly to the ghost nodes, for which a value *must* be defined. For ghost node values behind the domain boundaries, we use standard boundary conditions, which gives on the left boundary $\partial\Omega_l$

$$\frac{U_{0,j} + U_{1,j}}{2} = D_l(\boldsymbol{X}_{\frac{1}{2},j}) + \mathcal{O}(h^2)\,, \qquad \forall j \in [\![1,n]\!] \cap \partial\Omega_D\,, \tag{4a}$$

$$\frac{U_{1,j} - U_{0,j}}{h} = N_l(\boldsymbol{X}_{\frac{1}{2},j}) + \mathcal{O}(h^2)\,, \qquad \forall j \in [\![1,n]\!] \cap \partial\Omega_N\,. \tag{4b}$$

For ghost node values behind the immersed boundary, similar boundary conditions can be derived from Eq. (1b)–(1c) that also take into account the geometry of the immersed boundary; it is detailed in Sect. 2.2. Boundary conditions of all boundaries can be written in matrix form $(EU)_{i,j} = B_{i,j} + R_{i,j}, \forall (i,j) \in C_g$, where B contains boundary condition values, such as $D_l(\boldsymbol{X}_{\frac{1}{2},j})$ in Eq. (4a). The truncation term R will be refined in Sect. 2.2. Finally, both the discretized Poisson equation Eq. (3) and boundary conditions are assembled into the closed linear system

$$\big((L+E)U\big)_{i,j} = F_{i,j} + B_{i,j} + \mathcal{O}(h^2) + R_{i,j}\,, \qquad \forall (i,j) \in C_{ig}\,, \tag{5}$$

where $C_{ig} = C_i \cup C_g$. To write Eq. (5), we consider that $(LU)_{i,j} = F_{i,j} = 0$ over C_g and $(EU)_{i,j} = B_{i,j} = 0$ over C_i. This methodology can be applied to other problems by adding the part $EU = B + R$ to the considered equation. An example is given in Eq. (52a) of this article with the momentum equation.

Numerical simulations can be performed by dropping the truncation terms

$$\big((L+E)\hat{U}\big)_{i,j} = F_{i,j} + B_{i,j}\,, \qquad \forall (i,j) \in C_{ig}\,, \tag{6}$$

where $\hat{U}$ is the *numerical solution*. The discretization of the Poisson equation prevents ourselves from observing better than *second-order limiting behavior,* although this limitation is not due *a priori* to the immersed boundary method.

Implemented linear system arrays still have placeholders for outer node values due to the structured formulation, but their values should have no influence on the other node values. It is useful to set outer node values to a large value by setting both the diagonal and the right-hand side to, say, 10^{10}, as it helps bug detection in the implementation.

The numerical simulations presented in this document resort on the direct linear system solver *mumps* [12,13] or the iterative linear system solvers of *hypre* [10,14]. In the later case, we used GMRES as solver and SMG or BoomerAMG as preconditioner, unless specified otherwise.

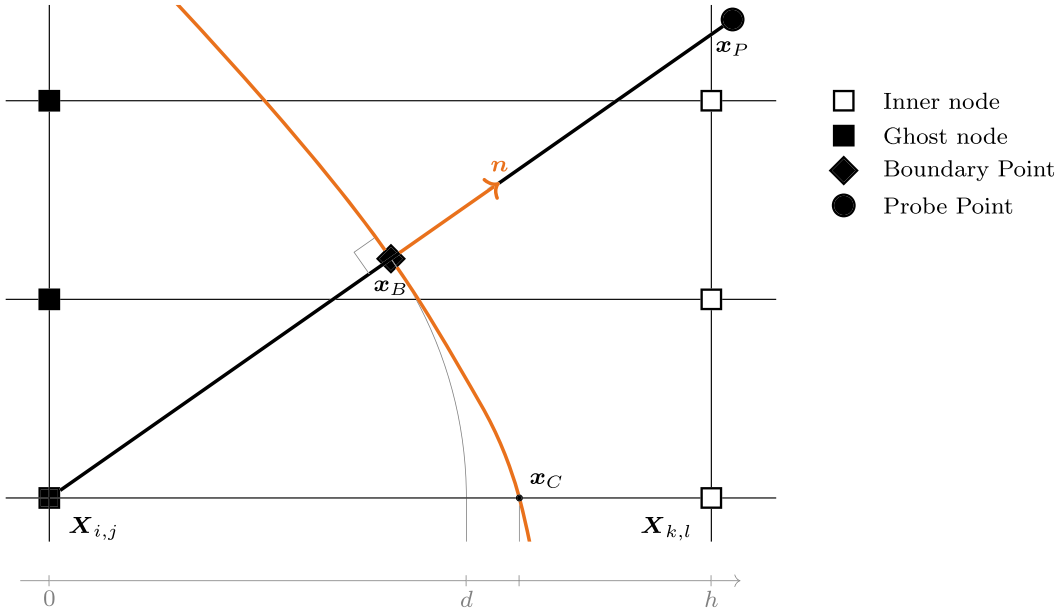


Fig. 2. Sketch defining the boundary point $\boldsymbol{x}_B$ and the probe point $\boldsymbol{x}_P$ with respect to the ghost node $\boldsymbol{X}_{i,j}$. The boundary point is the closest point of Γ with respect to $\boldsymbol{X}_{i,j}$, thus $[\boldsymbol{X}_{i,j}, \boldsymbol{x}_B]$ is aligned with $\boldsymbol{n}$. As $\boldsymbol{X}_{k,l}$ is an inner node, $\boldsymbol{x}_C$ is always defined. The axis at the bottom illustrates that $|\boldsymbol{x}_B - \boldsymbol{X}_{i,j}| \leq |\boldsymbol{x}_C - \boldsymbol{X}_{i,j}| < |\boldsymbol{X}_{k,l} - \boldsymbol{X}_{i,j}|$.

2.2. Immersed boundary conditions

In this section, we present two different definitions of immersed boundary conditions that were introduced by Mittal et al. [3], the *linear method* in this paper, and by Coco and Russo [4], the *direct method*.

Let us consider a ghost node $\boldsymbol{X}_{i,j}$ next to some part of the immersed boundary Γ. We define the *boundary point* $\boldsymbol{x}_B$ as the closest point of Γ with respect to $\boldsymbol{X}_{i,j}$; the scalar d is the distance between $\boldsymbol{x}_B$ and the ghost node, and the unit vector $\boldsymbol{n}$ defined in Sect. 2.1 happens to be the direction of $\boldsymbol{x}_B$ from the ghost node. We also define the *probe point* $\boldsymbol{x}_P$ as the symmetric of $\boldsymbol{X}_{i,j}$ with respect to $\boldsymbol{x}_B$. Fig. 2 illustrates these definitions.

The method used to obtain $\boldsymbol{x}_B$, $\boldsymbol{x}_P$, d, and $\boldsymbol{n}$ depends on how the immersed boundary is defined (by a parametrization, a level-set field, etc.). Immersed boundaries used in the numerical simulations presented in this document uses exact parametrizations or piecewise linear curves such as the variables can be evaluated up to machine epsilon.

2.2.1. Linear method [3]

The linear method builds boundary conditions using finite-differences along the *linear* segment $[\boldsymbol{X}_{i,j}, \boldsymbol{x}_P]$:

$$\frac{U_{i,j} + u_P}{2} = u_B + \mathcal{O}(d^2)\,, \tag{7a}$$

$$\frac{u_P - U_{i,j}}{2d} = \partial_{\boldsymbol{n}} u_B + \mathcal{O}(d^2)\,, \tag{7b}$$

where $u_P = u(\boldsymbol{x}_P)$, $u_B = u(\boldsymbol{x}_B)$, and $\partial_{\boldsymbol{n}} u_B = \partial_{\boldsymbol{n}} u(\boldsymbol{x}_B)$ are the *exact* field values and derivatives at the corresponding points. The right-hand side term $\mathcal{O}(d^2)$ is equivalent to $\mathcal{O}(h^2)$. Indeed, according to the definition of a ghost node Eq. (2) there exists an adjacent inner node $\boldsymbol{X}_{k,l}$, so there is an intersection point $\boldsymbol{x}_C$ between Γ and the line segment $[\boldsymbol{X}_{i,j}, \boldsymbol{X}_{k,l}]$; see Fig. 2. And, as $\boldsymbol{x}_B$ is the closest point of Γ from $\boldsymbol{X}_{i,j}$,

$$d = \|\boldsymbol{x}_B - \boldsymbol{X}_{i,j}\|_2 \leq \|\boldsymbol{x}_C - \boldsymbol{X}_{i,j}\|_2 < h. \tag{8}$$

The left-hand side Eq. (7) must use only (cell) node values to fit in the linear system Eq. (5), so the value u_P is expanded as a p-th order interpolation of surrounding node values:

$$u_P = \sum_{(k,l)\in I_P} \rho_{k,l} U_{k,l} + \mathcal{O}(h^p)\,, \tag{9}$$

where *interpolation coefficients* $\rho_{k,l}$ and the *set of interpolation indices* I_P will be defined in Sect. 2.3. The boundary conditions Eq. (1b)–(1c) can thus be discretized as

$$\tfrac{1}{2}U_{i,j} + \sum_{(k,l)\in I_P} \tfrac{1}{2}\rho_{k,l}U_{k,l} = D(\boldsymbol{x}_B) + \mathcal{O}(h^2) + \mathcal{O}(h^p)\,, \qquad \forall (i,j) \in C_g \cap \partial\Omega_D\,, \tag{10a}$$

$$\sum_{(k,l)\in I_P} \tfrac{1}{2d}\rho_{k,l}U_{k,l} - \tfrac{1}{2d}U_{i,j} = N(\boldsymbol{x}_B) + \mathcal{O}(h^2) + \mathcal{O}(h^{p-1})\,, \qquad \forall (i,j) \in C_g \cap \partial\Omega_N\,. \tag{10b}$$

A numerical approximation of the Poisson problem can be built using the linear method Eq. (10) to discretize the Dirichlet and Neumann boundary conditions. A second-order limiting behavior is obtained if *second-order interpolations* are used with Dirichlet boundary conditions, and if *third-order interpolations* are used with Neumann boundary conditions.

2.2.2. Direct method [4]

In this method, the probe point $\boldsymbol{x}_P$ is not used and the field value at the boundary point u_B is *directly* expanded as a p-th order interpolation, which gives for the Dirichlet boundary condition

$$\sum_{(k,l)\in I_B} \beta_{k,l}U_{k,l} = u_B + \mathcal{O}(h^p)\,, \tag{11}$$

where interpolation coefficients $\beta_{k,l}$ and the set of interpolation indices I_B will also be defined in Sect. 2.3. For the Neumann boundary condition, the field gradient can be obtained through algebraic differentiation of the interpolation polynomial:

$$\nabla u_B = \sum_{(k,l)\in I_B} (\partial_x\beta_{k,l}, \partial_y\beta_{k,l})U_{k,l} + \mathcal{O}(h^{p-1})\,, \tag{12}$$

where $\partial_x\beta_{k,l}$ and $\partial_y\beta_{k,l}$ are the interpolation coefficients of the polynomial gradient. Using $\partial_{\boldsymbol{n}}u_B = \nabla u_B \cdot \boldsymbol{n}$, the boundary conditions Eq. (1b)–(1c) can thus be discretized as

$$\sum_{(k,l)\in I_B} \beta_{k,l}U_{k,l} = D(\boldsymbol{x}_B) + \mathcal{O}(h^p)\,, \qquad \forall (i,j) \in C_g \cap \partial\Omega_D\,, \tag{13a}$$

$$\sum_{(k,l)\in I_B} (\partial_x\beta_{k,l}n_x + \partial_y\beta_{k,l}n_y)U_{k,l} = N(\boldsymbol{x}_B) + \mathcal{O}(h^{p-1})\,. \qquad \forall (i,j) \in C_g \cap \partial\Omega_N\,. \tag{13b}$$

A numerical approximation of the Poisson problem can be built using the direct method Eq. (13) to discretize the Dirichlet and Neumann boundary conditions. A second-order limiting behavior is obtained if *second-order interpolation* are used with Dirichlet boundary conditions, and if *third-order interpolation* are used with Neumann boundary conditions.

2.3. Interpolation

Interpolations introduced in Eq. (9), (11) are the last part to define the immersed boundary methods. Any interpolation method can be used as long as two constraints are met:

1. the set of interpolation nodes *must not* include any outer node,
2. Equation (9), (11) *must* include the ghost node value $U_{i,j}$.

(14)

The first constraint ensures that the linear system Eq. (5) is closed because no values are defined on outer nodes. The second constraint ensures that $U_{i,j}$ is defined by Eq. (9), (11). We will see later that this is not always the case for the direct method on rectangular cells.

This article uses Lagrange polynomials over the Cartesian set of nodes $\{\boldsymbol{X}_{k,l} = (X_k, Y_l)^\intercal \mid (k,l) \in I_\chi\}$, with $\chi = P$ for the probe point (Eq. (9)) and $\chi = B$ for the boundary point Eq. (11). The set of interpolation indices I_χ is defined by the indices of the two opposite corners of the Cartesian set. The first index (i_c, j_c) corresponds to the node surrounding $\boldsymbol{x}_\chi$ in the opposite direction given by $\boldsymbol{n}$, and the second index is $p-1$ nodes away from (i_c, j_c) in the direction given by $\boldsymbol{n}$. Using $\boldsymbol{n}$ extends X_χ towards the inner domain to conform to the first constraint. The formal definition of I_χ is as follows, and Fig. 3 shows examples:

- when $\boldsymbol{n} = (n_x, n_y)^\intercal$ is oriented to the top right *quadrant*, i.e. $n_x \geq 0$ and $n_y \geq 0$, then

$$i_c = \max\{k \mid X_k < x_\chi\}\,, \qquad j_c = \max\{l \mid Y_l < y_\chi\}\,, \qquad I_\chi = [\![i_c, i_c + p - 1]\!] \times [\![j_c, j_c + p - 1]\!]\,; \tag{15a}$$

- when $\boldsymbol{n}$ is oriented to the top left quadrant, i.e. $n_x < 0$ and $n_y \geq 0$, then

$$i_c = \min\{k \mid x_\chi < X_k\}\,, \qquad j_c = \max\{l \mid Y_l < y_\chi\}\,, \qquad I_\chi = [\![i_c - p + 1, i_c]\!] \times [\![j_c, j_c + p - 1]\!]\,; \tag{15b}$$

- when $\boldsymbol{n}$ is oriented to the bottom left quadrant, i.e. $n_x < 0$ and $n_y < 0$, then

$$i_c = \min\{k \mid x_\chi < X_k\}\,, \qquad j_c = \min\{l \mid y_\chi < Y_l\}\,, \qquad I_\chi = [\![i_c - p + 1, i_c]\!] \times [\![j_c + p - 1, j_c]\!]\,; \tag{15c}$$

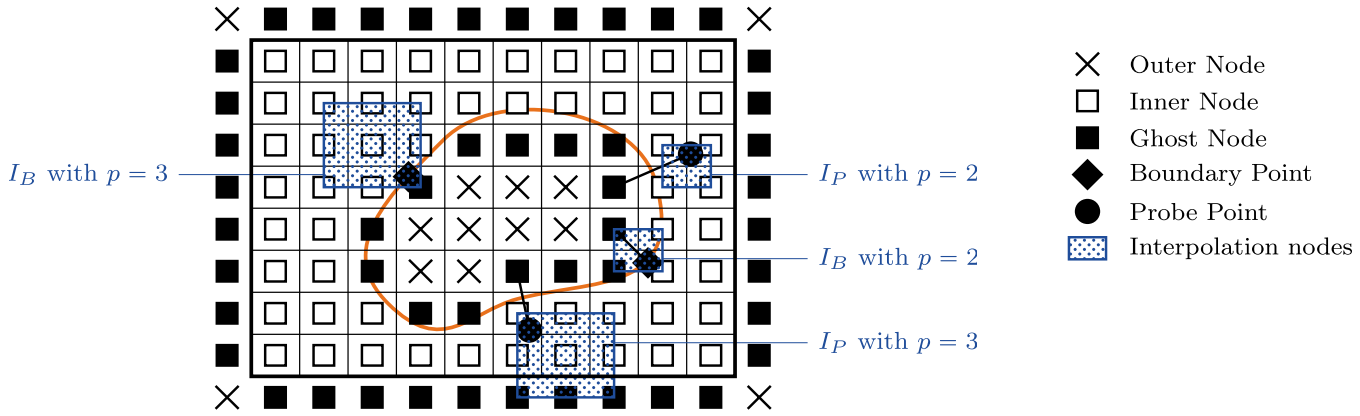


Fig. 3. Examples of sets of interpolation nodes for the grid sketched in Fig. 1b. At four ghost nodes, the boundary or probe point is shown, and the corresponding set of interpolation nodes is shown with a dot-filled rectangle.

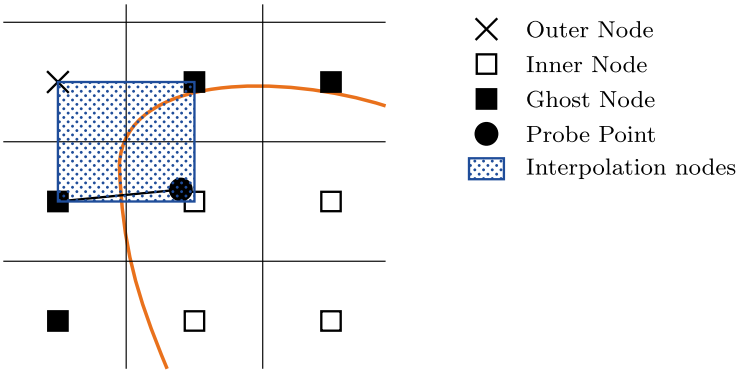


Fig. 4. Example of failing interpolation regions for both the linear and direct methods. Notice that this can happen only if the immersed boundary is curved towards the inner domain over a region of two cells width, i.e. the boundary is *hollow*.

- and when $\boldsymbol{n}$ is oriented to the bottom right quadrant, i.e. $n_x \geq 0$ and $n_y < 0$, then

$$i_c = \max\{k \mid X_k < x_\chi\}, \qquad j_c = \min\{l \mid y_\chi < Y_l\}, \qquad I_\chi = [\![i_c, i_c + p - 1]\!] \times [\![j_c + p - 1, j_c]\!]. \tag{15d}$$

This definition and Fig. 3 show that the region delimited by I_χ always contains $\boldsymbol{x}_\chi$, hence we always have interpolation and never extrapolation. Extending I_χ towards the direction given by $\boldsymbol{n}$ avoids I_χ to contain outer node indices. It is authorized to have other ghost nodes indices in I_χ as their values are defined by the corresponding immersed boundary condition. This results in couplings between the immersed boundary conditions that will be solved by the linear system Eq. (5).

However, even though Eq. (15) is designed to avoid outer nodes, one can still find a configuration where I_χ contains some outer nodes; see Fig. 4 for examples. This occurs near parts of the immersed boundary that are *hollow*, i.e. Γ is curved towards the inner domain, over a range of two cells width; this issue is called the *hollow case problem*. Several solutions exist in the literature: Coco and Russo [4] stretch the discretization stencil to fit into the local inner domain and to reach enough inner nodes for the desired accuracy; Mousel [15] takes all nodes in a broader region in which there are more than enough inner nodes, and uses a least-squares algorithm to determine interpolation coefficients. Using any of these methods increases the size of the discretization stencil as they use inner node values that are farther away from the ghost node. We do not use these methods in this article as we aim to keep the size of the discretization stencil tight. Instead, we resort on increasing the grid resolution as, for continuously differentiable immersed boundaries, the curvature with respect to the grid resolution decreases.

The second constraint is always met for the linear method as $U_{i,j}$ is directly present in the expression Eq. (10). This is not the case for the direct method Eq. (13), as $U_{i,j}$ is present through the interpolation only, which may not contain (i, j). Hopefully the ghost node correspond to the first corner node of I_B on square cells: $|x_B - X_i| < h = |X_{i+1} - X_i|$ and $|y_B - Y_j| < h = |Y_{j+1} - Y_j|$, so we have $(i, j) = (i_c, j_c)$, which ensures that the direct method always involves the ghost cell value. This is not true anymore for rectangular cells, see examples on Fig. 6 and Fig. 7 below.

3. Analysis of the discretization stencil

This section analyzes the discretization stencil of the immersed boundary method introduced in Sect. 2 in the general case of rectangular cells.

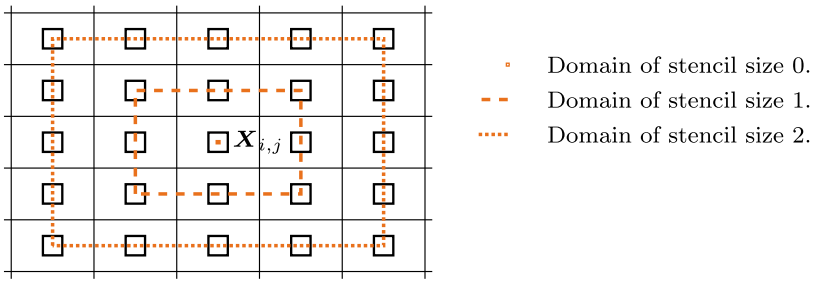


Fig. 5. Domains of stencil sizes 0, 1, 2 around $\boldsymbol{X}_{i,j}$. The domain of stencil size 0 is reduced to a single point $\{\boldsymbol{X}_{i,j}\}$.

3.1. Stencil size measurement

The matrix of the linear system that solves the Poisson problem with immersed boundaries Eq. (6) expands

$$(LU)_{i,j} = \sum_{(k,l)\in C_{ig}} L^{k,l}_{i,j} U_{k,l}\,, \qquad \forall (i,j) \in C_{ig}\,, \tag{16}$$

where $L^{k,l}_{i,j}$ are the scalar coefficients obtained from the discretization of the Poisson equation Eq. (3). Note that this expansion also applies for boundary conditions Eq. (10) or Eq. (13), i.e. over C_g, defining the scalar coefficients $E^{k,L}_{i,j}$. For an easier reading, the following does not write down expressions for E. The *stencil* of Eq. (16) is the set of indices $(k-i, l-j)$ whose coefficient $L^{k,l}_{i,j}$ is not zero. The *stencil size* $c_{i,j}$ of Eq. (16) is

$$c_{i,j} = \max\left\{\, |k-i|, |l-j| \mid \forall (k,l) \in C_{ig}, L^{k,l}_{i,j} \neq 0 \right\}, \qquad \forall (i,j) \in C_{ig}\,. \tag{17}$$

The expression "compact stencil" used in the literature means $c = 1$, and "non-compact stencil" means $c > 1$. From a metric perspective, the stencil size can be obtained with the *stencil norm,* defined as

$$\|\boldsymbol{x}\|_c = \max\left\{ \frac{|x|}{h_x}, \frac{|y|}{h_y} \right\}, \tag{18}$$

where $\boldsymbol{x} = (x,y)^\intercal \in \Omega$. The proof that $\boldsymbol{x} \mapsto \|\boldsymbol{x}\|_c$ being a norm comes easily with the equivalence property

$$\frac{1}{h}\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_c \leq \frac{1}{\mu}\|\boldsymbol{x}\|_\infty\,, \tag{19}$$

where $h = \max\{h_x, h_y\}$ and $\mu = \min\{h_x, h_y\}$. Noticing the equalities $X_k - X_i = h_x(k-i)$ and $Y_l - Y_j = h_y(l-j)$, the stencil size can be equivalently defined as

$$c_{i,j} = \max\left\{\, \|\boldsymbol{X}_{k,l} - \boldsymbol{X}_{i,j}\|_c \mid \forall (k,l) \in C_{ig}, L^{k,l}_{i,j} \neq 0 \right\}, \qquad \forall (i,j) \in C_{ig}\,, \tag{20}$$

that is the "grid" distance of the farthest node from $\boldsymbol{X}_{i,j}$ with $L^{k,l}_{i,j} \neq 0$. The region $\left\{ \boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{X}_{i,j}\|_c \leq c_{i,j} \right\}$ is the *domain of stencil size* $c_{i,j}$. Fig. 5 presents examples of such domains.

3.2. Stencil size for the Poisson problem

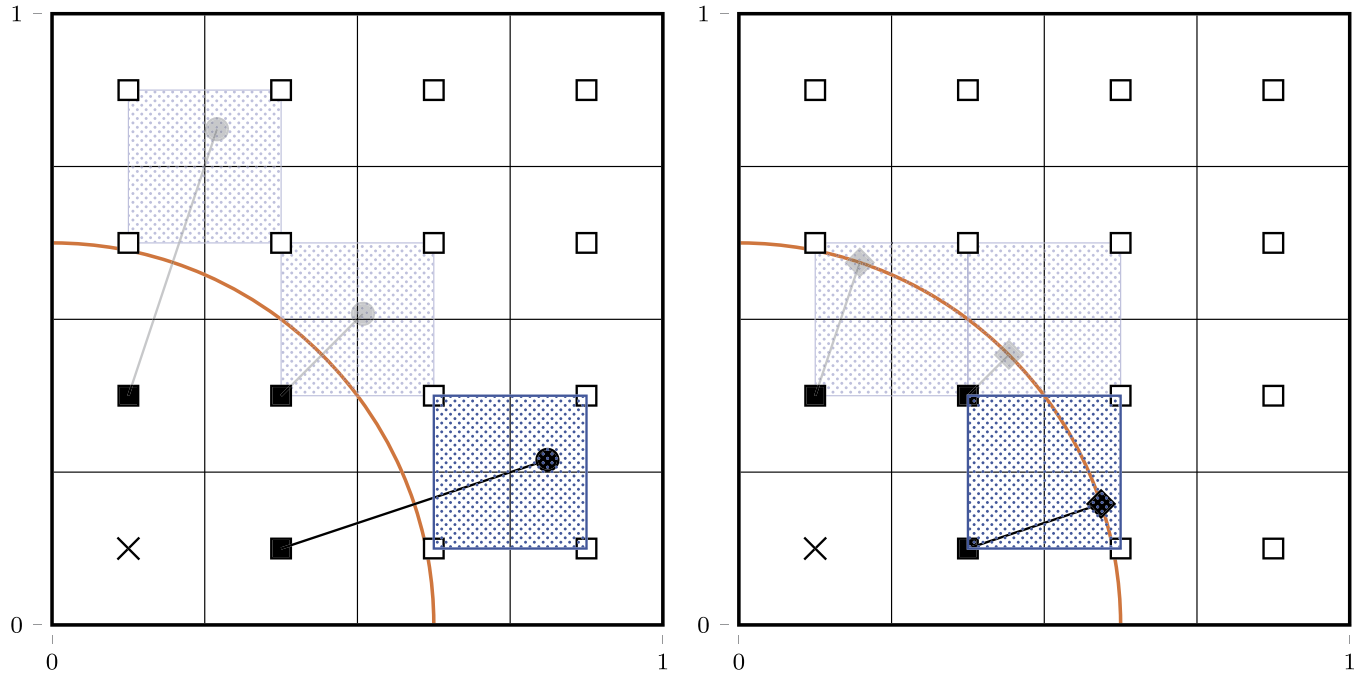
Let us review the stencil size for equations associated with each node types: at the inner nodes, the stencil size for the five-point stencil of the classical discretization of the Poisson equation Eq. (3) is 1; at the domain boundary conditions, Eq. (4) readily shows that the stencil size for these equations is also 1 for both Dirichlet and Neumann conditions. Finally, the stencil size for immersed boundary conditions Eq. (10) and Eq. (13) is determined by the farthest node included in I_χ, $\chi = P, B$, that is the *opposite corner node* defined in Sect. 2.3. Considering the ghost cell (i,j), we now determine the associated stencil size $c_{i,j}$:

- when $\boldsymbol{n}$ points to the top right quadrant, the opposite corner node is $(i_c + p - 1, j_c + p - 1)$ and

$$c_{i,j} = \|\boldsymbol{X}_{i_c+p-1, j_c+p-1} - \boldsymbol{X}_{i,j}\|_c = \|\boldsymbol{X}_{i_c+1, j_c+1} - \boldsymbol{X}_{i,j}\|_c + p - 2\,; \tag{21a}$$

- when $\boldsymbol{n}$ points to the top left quadrant, the opposite corner node is $(i_c - p + 1, j_c + p - 1)$ and

$$c_{i,j} = \|\boldsymbol{X}_{i_c-p+1, j_c+p-1} - \boldsymbol{X}_{i,j}\|_c = \|\boldsymbol{X}_{i_c-1, j_c+1} - \boldsymbol{X}_{i,j}\|_c + p - 2\,; \tag{21b}$$



(a) Linear method. (b) Direct method.

Fig. 6. Examples of sets of interpolation nodes I_P and I_B on square cells. Notations of Fig. 3 applies. For the linear method, equation associated with the highlighted ghost node has a stencil of 2 as the probe point is farther than one cell width. For the direct method with the same discrete problem, equation associated with the highlighted ghost node has a stencil of 1 as the boundary point is closer that one cell width.

- when $\boldsymbol{n}$ points to the bottom left quadrant, the opposite corner node is $(i_c - p + 1, j_c - p + 1)$ and

$$c_{i,j} = \|\boldsymbol{X}_{i_c-p+1,j_c-p+1} - \boldsymbol{X}_{i,j}\|_c = \|\boldsymbol{X}_{i_c-1,j_c-1} - \boldsymbol{X}_{i,j}\|_c + p - 2\,; \tag{21c}$$

- when $\boldsymbol{n}$ points to the bottom right quadrant, the opposite corner node is $(i_c + p - 1, j_c - p + 1)$ and

$$c_{i,j} = \|\boldsymbol{X}_{i_c+p-1,j_c-p+1} - \boldsymbol{X}_{i,j}\|_c = \|\boldsymbol{X}_{i_c+1,j_c-1} - \boldsymbol{X}_{i,j}\|_c + p - 2\,. \tag{21d}$$

The determination of $c_{i,j}$ is then deduced from the case $p = 2$ where I_χ contains just the four nodes surrounding $\boldsymbol{x}_\chi$;

- when $\boldsymbol{n}$ points to the right

$$x_\chi \le X_{i_c+1} < x_\chi + h_x, \qquad \text{leading to} \qquad \frac{|X_{i_c+1} - X_i|}{h_x} = \left\lceil \frac{|x_\chi - X_i|}{h_x} \right\rceil, \tag{22a}$$

- when $\boldsymbol{n}$ points to the left

$$x_\chi - h_x \le X_{i_c-1} < x_\chi, \qquad \text{leading to} \qquad \frac{|X_{i_c-1} - X_i|}{h_x} = \left\lceil \frac{|x_\chi - X_i|}{h_x} \right\rceil, \tag{22b}$$

- when $\boldsymbol{n}$ points to the top

$$y_\chi \le Y_{j_c+1} < y_\chi + h_y, \qquad \text{leading to} \qquad \frac{|Y_{j_c+1} - Y_j|}{h_y} = \left\lceil \frac{|y_\chi - y_j|}{h_y} \right\rceil, \tag{22c}$$

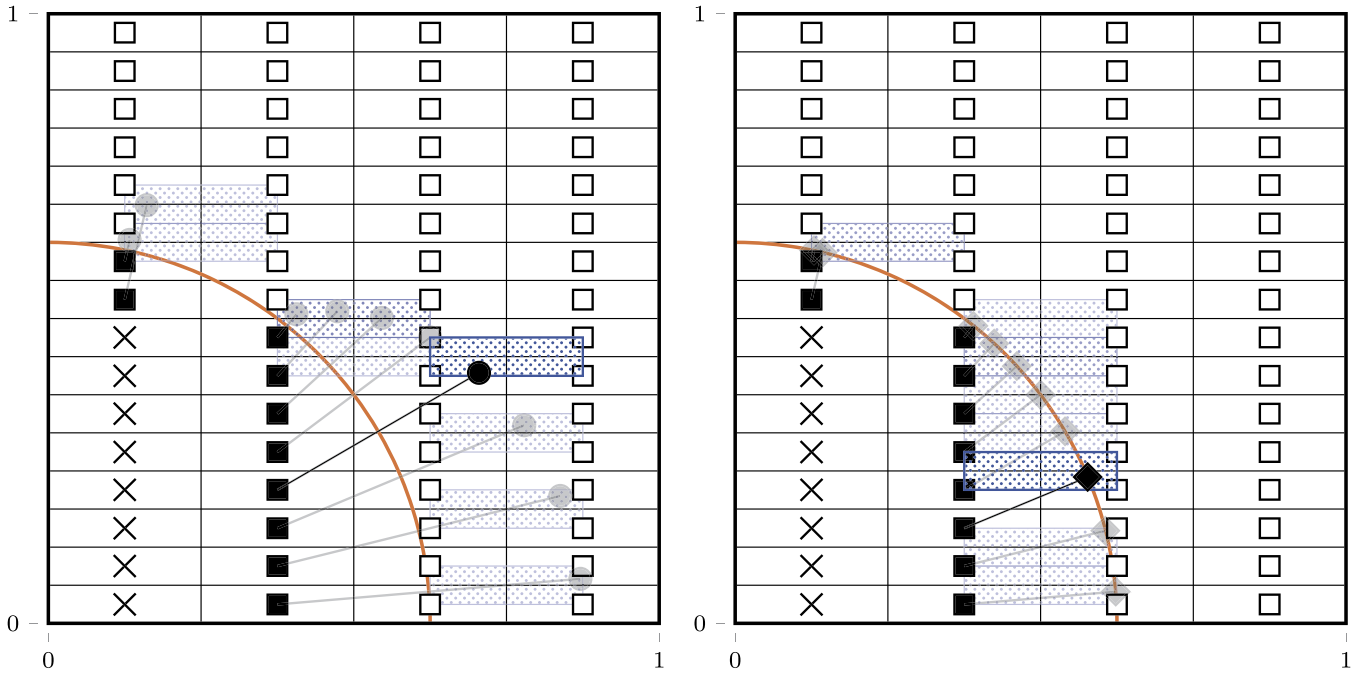
- when $\boldsymbol{n}$ points to the bottom

$$y_\chi - h_y \le Y_{j_c-1} < y_\chi, \qquad \text{leading to} \qquad \frac{|Y_{j_c-1} - Y_j|}{h_y} = \left\lceil \frac{|y_\chi - Y_j|}{h_y} \right\rceil, \tag{22d}$$

where $x \mapsto \lceil x \rceil$ is the *ceiling operator*. Therefore, we obtain for any direction of $\boldsymbol{n}$

$$c_{i,j} = \left\lceil \|\boldsymbol{x}_\chi - \boldsymbol{X}_{i,j}\|_c \right\rceil + p - 2\,. \tag{23}$$

We can now deduce $c_{i,j}$ from the position of $\boldsymbol{x}_\chi$ with respect to $\boldsymbol{X}_{i,j}$. Equation (8) shows that $\|\boldsymbol{x}_\chi - \boldsymbol{X}_{i,j}\|_2$ is bounded by the cell size. We can use this property again to find a useful approximation of Eq. (23) with the help of the equivalence property Eq. (19)



(a) Linear method. (b) Direct method.

Fig. 7. Examples of sets of interpolation nodes I_P and I_B on rectangular cells $a = 4$. Notations of Fig. 3 applies. For the linear method, equation associated with the highlighted ghost node has a stencil of 4, while for the direct method with the same discrete problem, equation associated with the highlighted ghost node has a stencil of 2. In the direct method, $U_{i,j}$ is not part of the set of interpolation nodes, and the boundary condition become ill posed.

Table 1
Stencil size for the Poisson problem with different immersed boundary conditions, immersed boundary methods, and different cell size ratios. Interpolation orders p are set to ensure second-order limiting behavior.

Cell size ratio	Method	Stencil size	
		Dirichlet $p = 2$	Neumann $p = 3$
1	Linear	2	3
	Direct	1	2
$a > 1$	Linear	$\lceil 2a \rceil$	$\lceil 2a \rceil + 1$
	Direct	n/a	n/a

$$\|\boldsymbol{x}_B - \boldsymbol{X}_{i,j}\|_c \leq \frac{1}{\mu}\|\boldsymbol{x}_B - \boldsymbol{X}_{i,j}\|_\infty \leq \frac{1}{\mu}\|\boldsymbol{x}_B - \boldsymbol{X}_{i,j}\|_2 \leq \frac{h}{\mu}, \qquad \|\boldsymbol{x}_P - \boldsymbol{X}_{i,j}\|_c \leq 2\|\boldsymbol{x}_B - \boldsymbol{X}_{i,j}\|_c \leq 2\frac{h}{\mu}. \tag{24}$$

This equation suggests that the stencil size increases as the *cell size ratio* $a = h/\mu$ increases, and examples presented in Figs. 6–7 show that it does increase. The figures present sets of interpolation nodes obtained with $p = 2$ and on very coarse grids around a circular immersed boundary. Fig. 6 uses a 4×4 mesh of squares cells. The stencil size is 1 for the direct method and 2 for the linear method due to the greater distance reached by the probe points with respect to the ghost node. Fig. 7 uses a 4×16 mesh of rectangular cells so $a = 4$. The highlighted ghost node has a stencil size of 2 for the direct method and 4 for the linear method. Indeed the stencil size has increased. More importantly for the direct method, (i, j) is no longer part of I_B and the second constraint of Eq. (14) is not verified anymore. Table 1 summarizes minimum stencil sizes to obtain a second-order limiting behavior for different cell size ratios and for Dirichlet and Neumann boundary conditions.

Conclusion. To reach second-order limiting behavior, this section showed that only the direct method is able to provide a stencil of size 1, i.e. a compact stencil, for the Dirichlet boundary condition. Moreover, it is not possible to expect a second-order limiting behavior when using Neumann boundary conditions on stencils of size 1 and for both methods; the stencil size must be at least equal to 2. On rectangular meshes, the direct method does not work anymore, and the stencil of the linear method increases as the cell size ratio a. Therefore band matrices become less suitable, and maybe inoperable, as more diagonals must be allocated as a increases. Next section proposes a method that reduces the stencil size to the one observed on square cells.

4. The Ghost Node Shifting Method

This section defines the Ghost Node Shifting Method introduced in this paper. It modifies the linear and direct methods described in Sect. 2.2 to reduces the stencil size discussed in Sect. 3.

At high cell size ratio, the closest point of Γ with respect to $\boldsymbol{X}_{i,j}$ may be several cells away, as in the example Fig. 7b. But, as shown in Fig. 2, there is at least one point, i.e. $\boldsymbol{x}_C$, which is less than one cell away from the ghost cell. The idea of the following method is to *shift* $\boldsymbol{X}_{i,j}$ towards $\boldsymbol{x}_C$ so that the new closest point will be closer to $\boldsymbol{x}_C$ and thus fewer cells away.

4.1. Ghost point

We consider a ghost cell (i,j) and its inner cell neighbor (k,l) as in Sect. 2.2. We define the *ghost point* $\boldsymbol{x}_G$ as

$$\boldsymbol{x}_G = \boldsymbol{x}_C - s\boldsymbol{e}\,, \qquad s = \min\left\{\mu, \|\boldsymbol{x}_C - \boldsymbol{X}_{i,j}\|_2\right\}, \qquad \boldsymbol{e} = \begin{cases} (+1,0)^\intercal & \text{if } (k,l) = (i+1,j)\,, \\ (-1,0)^\intercal & \text{if } (k,l) = (i-1,j)\,, \\ (0,+1)^\intercal & \text{if } (k,l) = (i,j+1)\,, \\ (0,-1)^\intercal & \text{if } (k,l) = (i,j-1)\,, \end{cases} \tag{25}$$

recalling that $\mu = \min\{h_x, h_y\}$. The unit vector $\boldsymbol{e}$ gives the direction of $\boldsymbol{x}_C$ from $\boldsymbol{x}_G$ and the definition of s ensures that the distance between the two points is at most μ. Fig. 8 illustrates the definition of $\boldsymbol{x}_G$ for $(k,l) = (i+1,j)$. The situation from the point of view of $\boldsymbol{x}_G$ is similar to the case of a square cell of size μ, which fits in the cell (i,j). Now the quantities $\boldsymbol{x}_B$, $\boldsymbol{x}_P$, d, $\boldsymbol{n}$ are redefined as in Sect. 2.2, but using $\boldsymbol{x}_G$ instead of $\boldsymbol{X}_{i,j}$. The modified quantities are noted $\boldsymbol{x}_{B'}$, $\boldsymbol{x}_{P'}$, d', $\boldsymbol{n}'$ and Fig. 8 also shows these redefinitions.

The representation of the immersed boundary becomes important here. If Γ is represented by an exact parametrization or a piecewise linear curve, evaluations of d' and $\boldsymbol{n}'$ can be evaluated up to machine epsilon; if Γ is represented by a discretized level-set field, some sort of interpolation must be considered. In the numerical simulations presented in this document, d' and $\boldsymbol{n}'$ can be evaluated up to machine epsilon.

4.2. Shifted linear and direct methods

The steps of the linear method Sect. 2.2.1 are done again replacing $U_{i,j}$ by $u_G = u(\boldsymbol{x}_G)$

$$\frac{u_G + u_{P'}}{2} = u_{B'} + \mathcal{O}(d'^2)\,, \tag{26a}$$

$$\frac{u_{P'} - u_G}{2d'} = \partial_{\boldsymbol{n}} u_{B'} + \mathcal{O}(d'^2)\,. \tag{26b}$$

As u_G is not a node value, another p-th order interpolation is introduced:

$$u_G = \sum_{(k,l)\in I_G} \gamma_{k,l} U_{k,l} + \mathcal{O}(h^p)\,, \tag{27}$$

where interpolation coefficients $\gamma_{k,l}$ and the set of interpolation indices I_G are computed as described in Sect. 2.3. The fully discretized equations are then

$$\sum_{(k,l)\in I_G} \frac{1}{2}\gamma_{k,l} U_{k,l} + \sum_{(k,l)\in I_{P'}} \frac{1}{2}\rho'_{k,l} U_{k,l} = D(\boldsymbol{x}_{B'}) + \mathcal{O}(h^2) + \mathcal{O}(h^p)\,, \qquad \forall (i,j)\in C_g\,, \tag{28a}$$

$$\sum_{(k,l)\in I_{P'}} \frac{1}{2d'}\rho'_{k,l} U_{k,l} - \sum_{(k,l)\in I_G} \frac{1}{2d'}\gamma_{k,l} U_{k,l} = N(\boldsymbol{x}_{B'}) + \mathcal{O}(h^2) + \mathcal{O}(h^{p-1})\,, \qquad \forall (i,j)\in C_g\,, \tag{28b}$$

where $\rho'_{k,l}$ are the interpolation coefficients for P'. Like the non-shifted Eq. (10), a numerical approximation of the Poisson problem can be built using the shifted linear method Eq. (28), and a second-order limiting behavior is obtained if second-order interpolations are used with Dirichlet boundary conditions, and if third-order interpolations are used with Neumann boundary conditions.

The direct method Sect. 2.2.2 can be applied seamlessly.

$$\sum_{(k,l)\in I_{B'}} \beta'_{k,l} U_{k,l} = D(\boldsymbol{x}_{B'}) + \mathcal{O}(h^p)\,, \qquad \forall (i,j)\in C_g \cap \partial\Omega_D\,, \tag{29a}$$

$$\sum_{(k,l)\in I_{B'}} (\partial_x \beta'_{k,l} n_x + \partial_y \beta'_{k,l} n_y) U_{k,l} = N(\boldsymbol{x}_{B'}) + \mathcal{O}(h^{p-1})\,, \qquad \forall (i,j)\in C_g \cap \partial\Omega_N\,, \tag{29b}$$

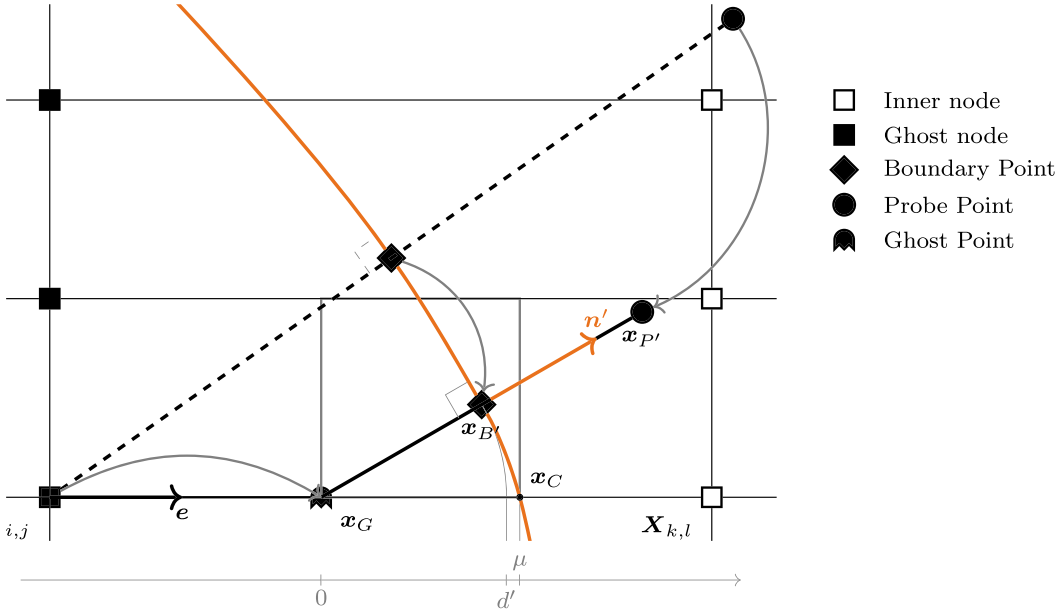


Fig. 8. The Ghost Node Shifting Method applied to the example Fig. 2. The ghost point $\boldsymbol{x}_G$, initially placed at $\boldsymbol{X}_{i,j}$, is shifted towards $\boldsymbol{X}_{k,l}$. As a result, boundary and probe points computed from $\boldsymbol{x}_G$ are closer to the cell (i, j). The actual position of $\boldsymbol{x}_G$ is at the corner of the gray square of size μ.

where $\beta'_{k,l}$ are the interpolation coefficients for B', and $\partial_x\beta'_{k,l}$ and $\partial_y\beta'_{k,l}$ are the interpolation coefficients of the polynomial gradient for B'. The effects of the Ghost Node Shifting Method affects the equations through $\boldsymbol{x}_{B'}$ and $\boldsymbol{n}'$ only. Like the non-shifted Eq. (13), a numerical approximation of the Poisson problem can be built using the shifted direct method Eq. (29), and a second-order limiting behavior is obtained if second-order interpolations are used with Dirichlet boundary conditions, and if third-order interpolations are used with Neumann boundary conditions.

4.3. Stencil size of the shifted methods

The discussion of Sect. 3.2 applies here, and Eq. (23) holds. In this section, we look for a new bounding of $\|\boldsymbol{x}_\chi - \boldsymbol{X}_{i,j}\|_c$, $\chi = P, B$ to provide a better approximation of Eq. (23). Let's start by developing the norm

$$\|\boldsymbol{x}_\chi - \boldsymbol{X}_{i,j}\|_c \leq \max\left\{\frac{|x_\chi - x_G| + |x_G - x_i|}{h_x}, \frac{|y_\chi - y_G| + |y_G - y_j|}{h_y}\right\}. \tag{30}$$

The $\boldsymbol{x}_G - \boldsymbol{X}_{i,j}$ part depends on the location of the neighbor inner cell

$$\begin{cases} |x_G - x_i| < h_x - \mu\,, |y_G - y_j| = 0\,, & \text{if } (k,l) = (i+1, j)\,, \\ |x_G - x_i| < h_x - \mu\,, |y_G - y_j| = 0\,, & \text{if } (k,l) = (i-1, j)\,, \\ |x_G - x_i| = 0\,, \quad |y_G - y_j| < h_y - \mu\,, & \text{if } (k,l) = (i, j+1)\,, \\ |x_G - x_i| = 0\,, \quad |y_G - y_j| < h_y - \mu\,, & \text{if } (k,l) = (i, j-1)\,. \end{cases} \tag{31}$$

The $\boldsymbol{x}_\chi - \boldsymbol{x}_G$ part exploits its similarity with a square cell of size μ

$$\|\boldsymbol{x}_{B'} - \boldsymbol{x}_G\|_\infty \leq \|\boldsymbol{x}_{B'} - \boldsymbol{x}_G\|_2 \leq \|\boldsymbol{x}_C - \boldsymbol{x}_G\|_2 \leq \mu\,, \tag{32a}$$

and, by symmetry of the probe point,

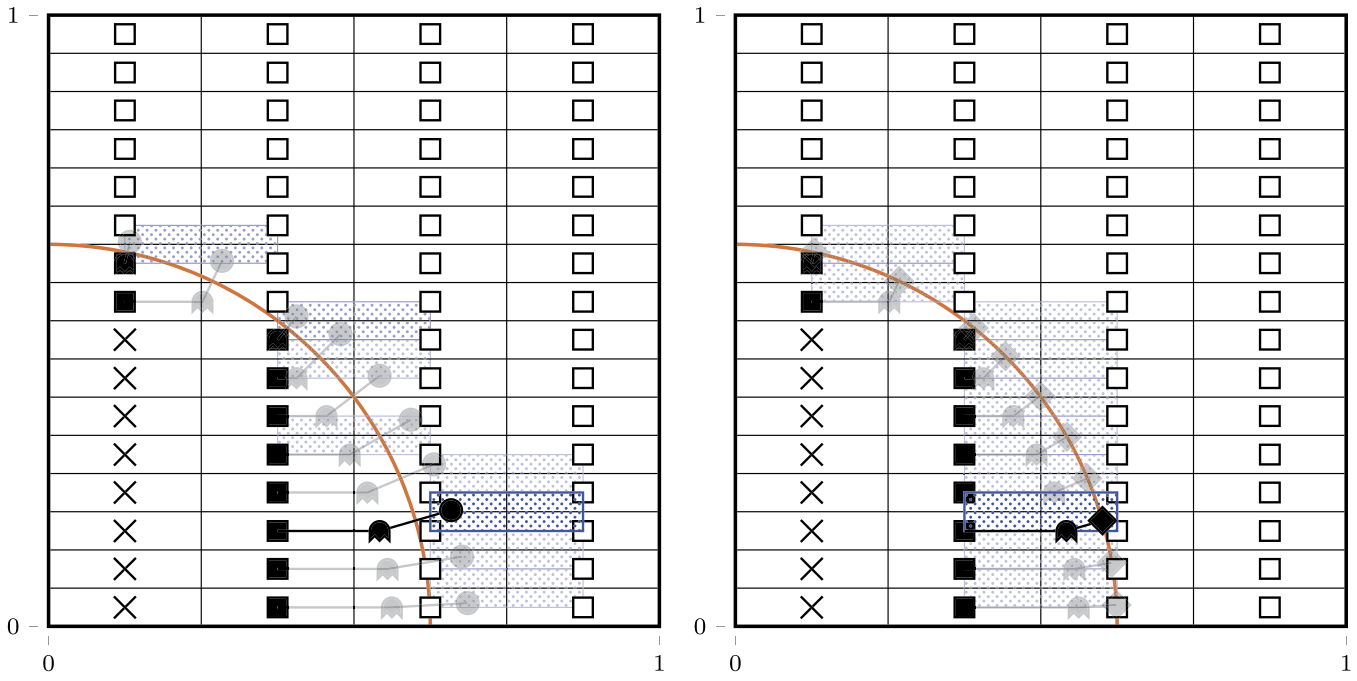
$$\|\boldsymbol{x}_{P'} - \boldsymbol{x}_G\|_\infty \leq 2\mu\,. \tag{32b}$$

The absolute values in Eq. (30) can then be replaced by these upper bounds. If $(k,l) = (i \pm 1, j)$:

$$\|\boldsymbol{x}_{P'} - \boldsymbol{X}_{i,j}\|_c \leq \max\left\{\frac{h_x - \mu + 2\mu}{h_x}, \frac{2\mu}{h_y}\right\} \leq 2\,, \qquad \|\boldsymbol{x}_{B'} - \boldsymbol{X}_{i,j}\|_c \leq \max\left\{\frac{h_x - \mu + \mu}{h_x}, \frac{\mu}{h_y}\right\} \leq 1\,, \tag{33a}$$

and if $(k,l) = (i, j \pm 1)$:

$$\|\boldsymbol{x}_{P'} - \boldsymbol{X}_{i,j}\|_c \leq \max\left\{\frac{2\mu}{h_x}, \frac{h_y - \mu + 2\mu}{h_y}\right\} \leq 2\,, \qquad \|\boldsymbol{x}_{B'} - \boldsymbol{X}_{i,j}\|_c \leq \max\left\{\frac{\mu}{h_x}, \frac{h_y - \mu + \mu}{h_y}\right\} \leq 1\,. \tag{33b}$$



(a) Shifted linear method. (b) Shifted direct method.

Fig. 9. Examples of sets of interpolation nodes $I_{P'}$ and $I_{B'}$ on rectangular cells $a = 4$. Notations of Fig. 3 applies. Using the Ghost Node Shifting Method on the same discrete problem than Fig. 9, equation associated with the highlighted ghost node has a reduced stencil of 2 for the linear method, and 1 for the direct method. In the direct method, $U_{i,j}$ is now part of the set of interpolation nodes.

Table 2
Stencil size for the Poisson problem with different immersed boundary conditions, immersed boundary methods, and different cell size ratios. Interpolation orders p are set to ensure second-order limiting behavior.

Cell size ratio	Method	Stencil size	
		Dirichlet $p = 2$	Neumann $p = 3$
1	Linear	2	3
	Direct	1	2
$a > 1$	Linear	$\lceil 2a \rceil$	$\lceil 2a \rceil + 1$
	Direct	n/a	n/a
$a > 1$	Shifted Linear	2	3
	Shifted Direct	1	2

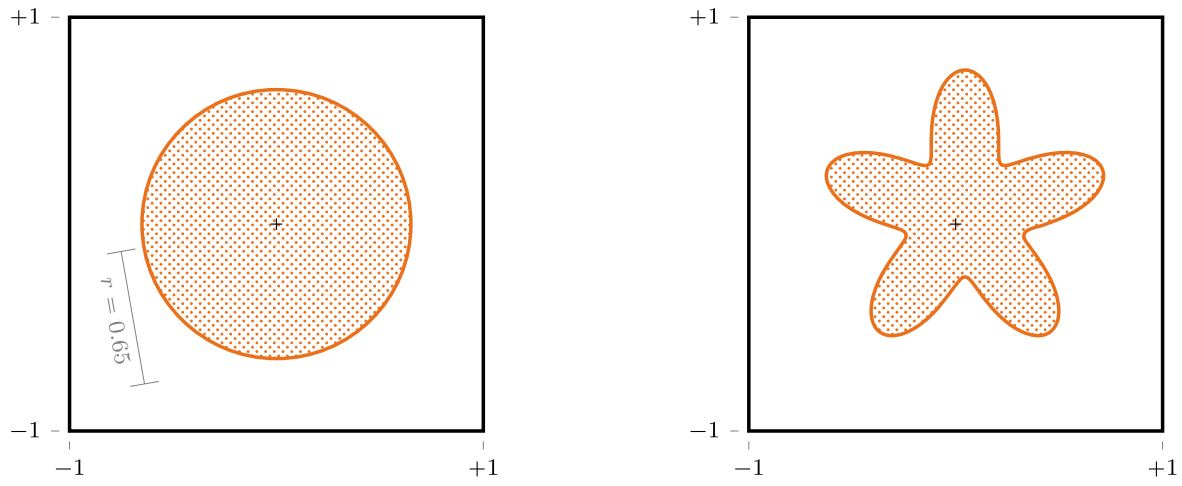
Contrary to Eq. (24), these bounds do not depend on the cell size ratio. The example presented in Sect. 3.2 (Figs. 6–7) is completed here by Fig. 9, which uses the shifted methods. The highlighted ghost node now has a stencil of 1 for the direct method and 2 for the linear method. Table 2 summarizes minimum stencil sizes to obtain a second-order limiting behavior and for different cell size ratios.

Conclusion. This section showed that it is possible to build immersed boundary conditions with a stencil size independent of the cell size ratio. The stencil size for rectangular cells is the same than the stencil size for square cells, which is the lowest size to be expected. With the Ghost Node Shifting Method, it is also possible to use the direct method on rectangular cells. This result is particularly useful to use band-matrix-limited linear system solvers such as in the *hypre* library. The next section present numerical results obtained for the Poisson problem.

5. Numerical simulations with the Poisson problem

This section discusses simulations of the Poisson problem solved with the Ghost Node Shifting Method and compare them with their classic counterparts.

Diagnostics. Simulations consists of convergence studies. As analytical solutions are available, the *error field* $\hat{U} - U$ can be computed at inner nodes, and $\mathcal{L}^2$ norm and $\mathcal{L}^\infty$ norm are used for comparison. These norms are



(a) Circular immersed boundary. (b) Flower-shaped immersed boundary.

Fig. 10. Computational domain and immersed boundary used for numerical simulations of Sect. 5. The immersed boundary is shown by a thick colored line, and the outer domain is shaded by dots.

$$\mathcal{L}^2(\hat{U}-U)=\sqrt{\sum_{(i,j)\in C_i}|\hat{U}_{i,j}-U_{i,j}|^2V_{i,j}}\,,\qquad \mathcal{L}^\infty(\hat{U}-U)=\max_{(i,j)\in C_i}|\hat{U}_{i,j}-U_{i,j}|\,, \tag{34}$$

where $V_{i,j}$ is the volume of the inner portion of the cell (i,j). The numerical evaluation of $V_{i,j}$ is subject to errors due to the approximation in the representation of the immersed boundary, and it must be carefully done. We found that using cell-wise linear approximation were inaccurate and unreliable to evaluate the order of convergence. This happens because errors in $V_{i,j}$ computed this way were of the same order that $|\hat{U}_{i,j}-U_{i,j}|^2$. Instead we use a sub-sampling method as follows: first, take 400×400 sampling points evenly spread in the cell; then evaluate the ratio of sampling points included in Ω_i; finally approximate the cell volume by multiplying $h_x\times h_y$ to this ratio. With this method, errors in $V_{i,j}$ are below 10^{-6} for all runs.

Computational domain. The common computational domain for all simulations in this section is the square $\Omega=[-1,+1]^2$, with Dirichlet boundary conditions at the x-axis boundaries and Neumann boundary conditions at the y-axis boundaries. In the first set of simulations, the immersed boundary Γ is a circle of center $(0,0)^\intercal$ and of radius 0.65; in the second set of simulations the immersed boundary Γ has a 5-branch flower shape defined by the parametrization

$$\vartheta\mapsto(0.02\sqrt{5},0.02\sqrt{5})^\intercal+\big(0.5+0.2\sin(5\vartheta)\big)e_r\,,\qquad\forall\vartheta\in[0,2\pi[\,. \tag{35}$$

This shape has also been used in [16,17]. In both immersed boundaries, the inner part Ω_i is outside, as illustrated in Fig. 10.

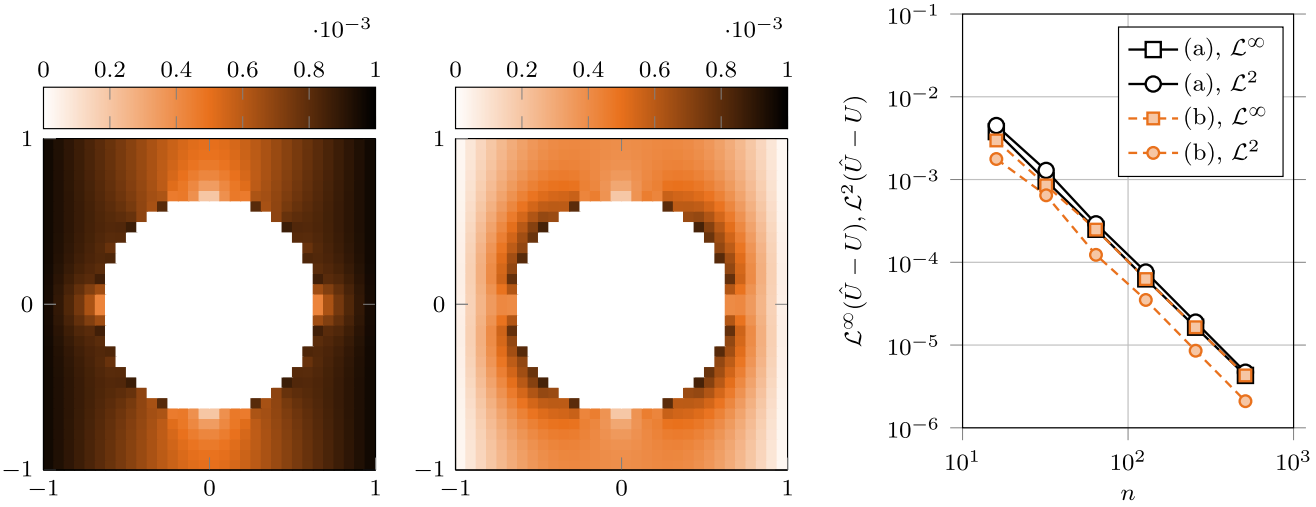
Grids. We consider three mesh series with cell size ratio a of 1, 2.8, and 7.6. The first grid of each series has a cell size $m\times n$ of 16×16, 28×10, and 76×10, respectively. The next grid of each series is obtained by multiplying the preceding m and n by two in all directions. Thus convergence study can be performed on each of the grid series.

Numerical correction. Differences between numerical and exact solutions arise from the truncation terms $R_{i,j}=F_{i,j}-(LU)_{i,j}$ of the Poisson equation Eq. (3), the domain boundary conditions Eq. (4), and the immersed boundary conditions Eq. (10), (13), (28), (29) altogether. In the resulting error field $\hat{U}-U$, differences from all sources are fused together by diffusion, and numerical errors resulting from immersed boundary conditions alone cannot be readily seen. As an example, Fig. 11a shows the error field for the first numerical run presented in Sect. 5.1: the largest errors are both near the domain boundary and near the immersed boundary, which is coherent with the magnitude of the truncation terms at $\boldsymbol{x}=(1,0)^\intercal$ and $\boldsymbol{x}=(0.65,0)^\intercal$

$$r_{h_x,h_y}(1,0)=-\frac{h_x^2}{4},\qquad r_{h_x,h_y}(0.65,0)=-\Big((1-\theta)\theta^2+(1-\theta)^2\theta\Big)\frac{h_x^2}{2},\quad\text{with }\theta\in[0,1], \tag{36}$$

where $r_{h_x,h_y}(\boldsymbol{x})$ is the truncation term taken at point $\boldsymbol{x}$. Equation (40) has been used for $r_{h_x,h_y}(1,0)$ and Eq. (41) has been used for $r_{h_x,h_y}(0.65,0)$.

As we are primarily interested on errors caused by the immersed boundary conditions, we decided to cancel error sources that are not related to immersed boundaries. To do this, we compute $R_{i,j}$ from the exact solution and we add it to the right-hand side of the system for all but immersed boundary condition. We call this the *numerical correction*. Fig. 11b shows effects on the error field when the numerical correction is applied: remaining errors are clearly located near



(a) Error field $|\hat{U} - U|$ computed without numerical correction.

(b) Error field $|\hat{U} - U|$ computed with numerical correction.

(c) Convergence study Fig. 12 of error field whose first grids correspond to (a, b).

Fig. 11. (a, b) Effects of numerical correction on the error fields $|\hat{U} - U|$ for the problem Sect. 5.1 on the 16×16 grid and with the non-shifted direct method. (c) Effects of numerical correction on the convergence plot Fig. 12 with $a = 1$ and with the non-shifted direct method. A second-order limiting behavior is shown for the error field both with and without numerical correction, but the norm with the largest error is not the same: $\mathcal{L}^2$ without numerical correction and $\mathcal{L}^\infty$ with numerical correction.

immersed boundaries. Finally, Fig. 11c compares convergence results in $\mathcal{L}^\infty$ and $\mathcal{L}^2$ norms for the same computations: while the second-order behavior is present in the two computations, errors are divided by $\frac{5}{2}$ in the $\mathcal{L}^2$ norm when numerical correction is applied.

5.1. Dirichlet immersed boundary condition for the circular interface

Analytical solution. A solution of the Poisson equation is given by

$$u(x, y) = (1 + x)^2\,, \qquad f(x, y) = -2\,, \qquad \forall (x, y) \in \Omega_i\,, \tag{37}$$

provided consistent boundary conditions

$$u(-1, y) = 0\,, \qquad u(+1, y) = 4\,, \qquad \forall y \in [-1, +1]\,, \tag{38a}$$

$$\frac{\partial u}{\partial y}(x, -1) = 0\,, \qquad \frac{\partial u}{\partial y}(x, +1) = 0\,, \qquad \forall x \in [-1, +1]\,, \tag{38b}$$

$$u(x, y) = (1 + x)^2\,, \qquad \forall (x, y) \in \Gamma\,. \tag{38c}$$

The truncation terms for this particular solution are

$$F_{i,j} - \frac{U_{i-1,j} - U_{i,j}}{h_x^2} - \frac{U_{i+1,j} - U_{i,j}}{h_x^2} - \frac{U_{i,j-1} - U_{i,j}}{h_y^2} - \frac{U_{i,j+1} - U_{i,j}}{h_y^2} = 0\,, \quad \forall (i, j) \in C_i\,, \tag{39}$$

$$\begin{aligned} &0 - \frac{U_{0,j} + U_{1,j}}{2} = -\tfrac{1}{4}h_x^2\,, \quad 4 - \frac{U_{m,j} + U_{m+1,j}}{2} = -\tfrac{1}{4}h_x^2\,, \quad \forall j \in [\![1, n]\!]\,, \\ &0 - \frac{U_{i,1} - U_{i,0}}{h_y} = 0\,, \qquad 0 - \frac{U_{i,n+1} - U_{i,n}}{h_y} = 0\,, \qquad \forall i \in [\![1, m]\!]\,. \end{aligned} \tag{40}$$

Therefore only the Dirichlet boundary conditions at the x-axis boundaries will be numerically corrected.

Results. Numerical solutions $\hat{U}$ are computed using the linear and the direct methods for the isotropic series, and the non-shifted linear (NL) method, the shifted linear (SL) method, and the shifted direct (SD) method for the two anisotropic series. Second-order interpolations ($p = 2$) are used to build immersed boundary conditions. Fig. 12 shows $\mathcal{L}^2$ and $\mathcal{L}^\infty$ norms of the error field $\hat{U} - U$. As expected, a second-order limiting behavior is obtained with all methods.

The figure also shows that the SD method has less errors than the NL method. The $\mathcal{L}^2$ norm of the SL method is similar to the $\mathcal{L}^2$ norm of the SD method, whereas the $\mathcal{L}^\infty$ norm of the SL method is as large as the NL method when $a = 2.8$, and changes when $a = 7.6$. Fig. 13 shows the error field for the 608×80 grid, e.g. when the $\mathcal{L}^\infty$ norm of the SL method does not decrease as expected: large error occurs around four nodes only. On the close up Fig. 13a, ghost cells are filled according to the compactness: white if $c_{i,j} = 1$, and black if $c_{i,j} = 2$. One can readily see that large errors occur near nodes

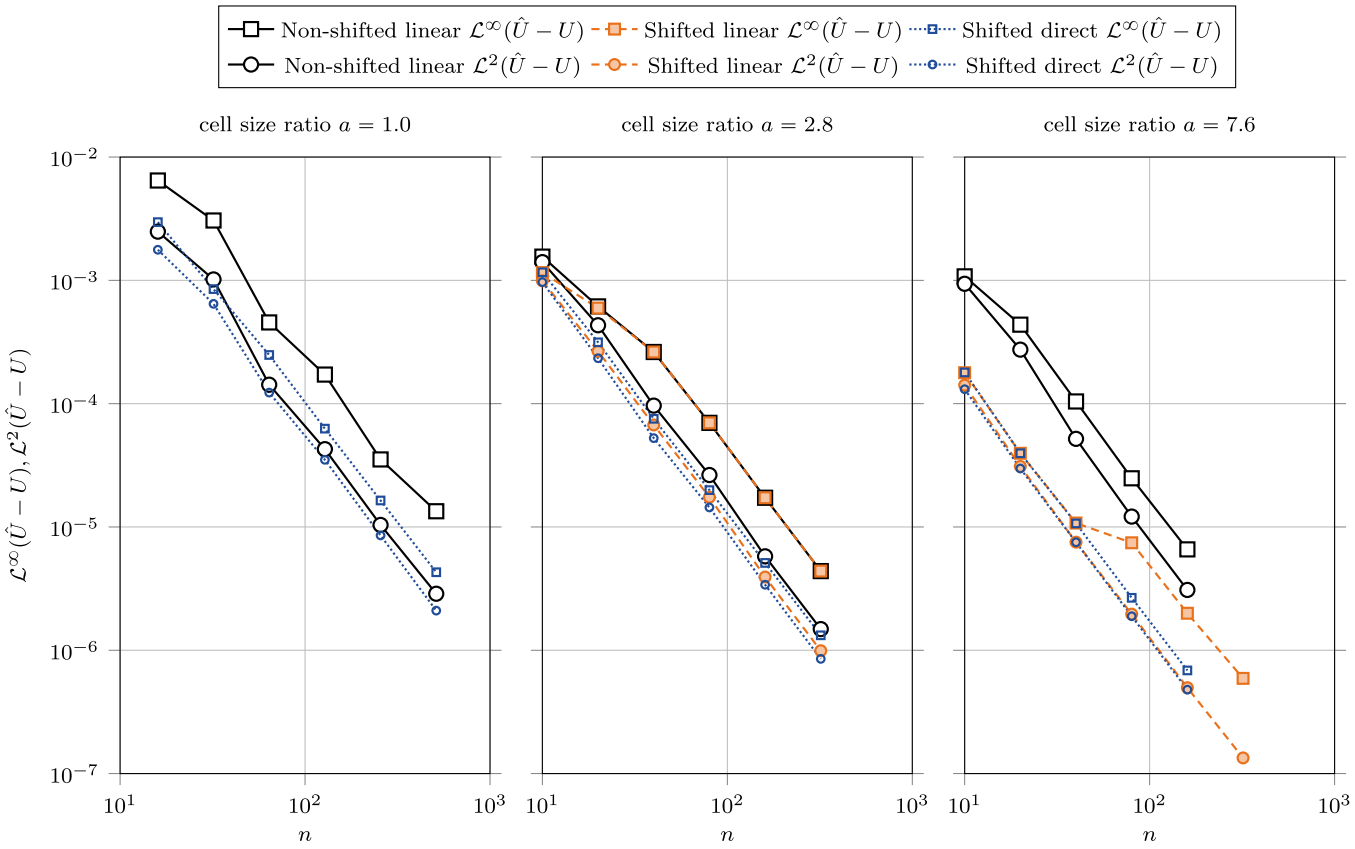
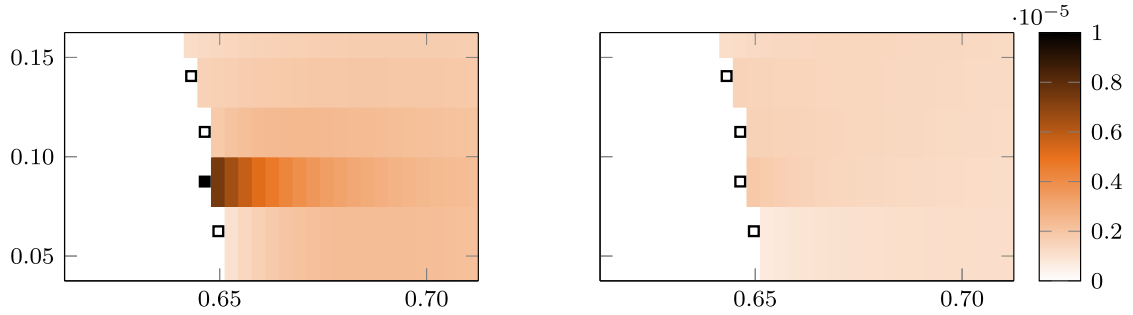


Fig. 12. Norms of the solution error $\hat{U} - U$ of the Poisson problem for Dirichlet immersed boundary condition on the circular interface Sect. 5.1. Each plot represents a mesh series. Solutions have been computed with the non-shifted linear, shifted linear, and shifted direct methods, except for the shifted linear method on the mesh series $a = 1.0$ as it is equivalent to the non-shifted linear method. The non-shifted direct did not converge when $a > 1.0$ as expected in Sect 3.2 and in Fig. 7. The x-axis represents $1/h = 1/\max\{h_x, h_y\} = \min\{n_x, n_y\}$, which is always equal to n in this section.



(a) Shifted linear method (b) Shifted direct method

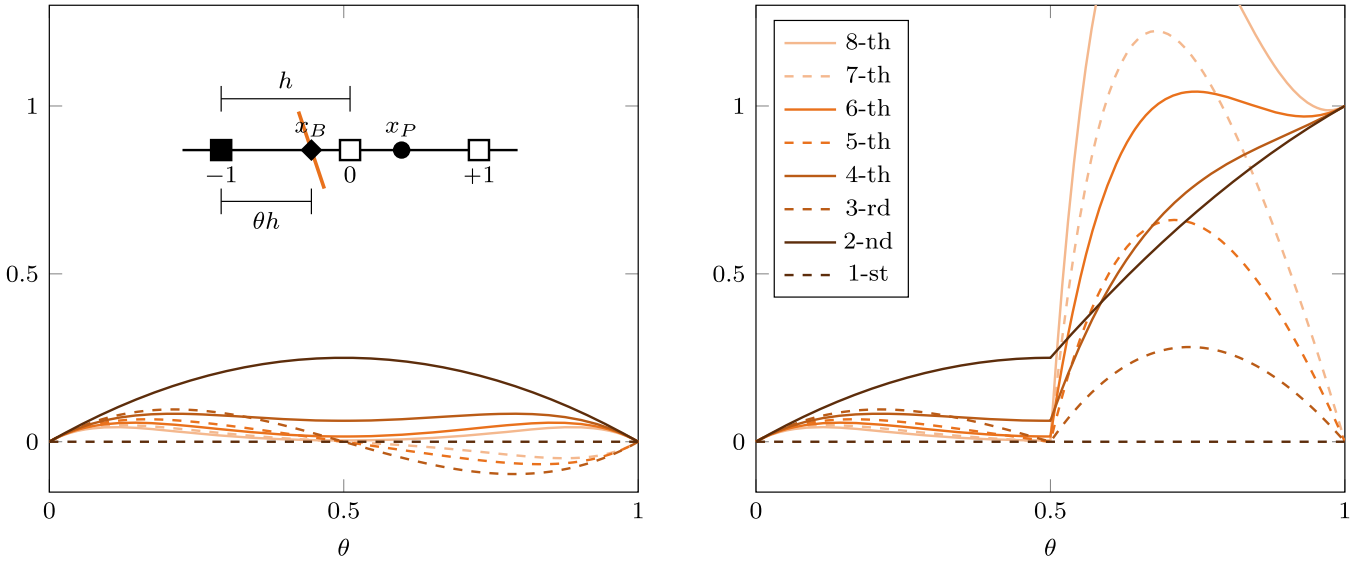
Fig. 13. Error fields $|\hat{U} - U|$ for the solution of Poisson problem with Dirichlet boundary conditions described in Sect. 5.1. Open symbols indicates ghost cell with a stencil size of 1, and closed symbols indicates ghost cells with a stencil size of 2.

with $c_{i,j} = 2$. By comparison, Fig. 13b shows the corresponding results with the SD method: as $c_{i,j} = 1$ everywhere, large errors have disappeared. This feature calls for a finer analysis of the truncation error.

Taylor series of the truncation error. Let's consider a one-dimensional grid containing only three grid nodes, located at: $x_{-1} = -h$, $x_0 = 0$, and $x_{+1} = +h$. An immersed boundary crosses the grid at $x_B = x_{-1} + \theta h$, with $\theta \in [0, 1]$, as shown in the diagram Fig. 14a. Using Taylor series of a scalar field u expanded at x_B, the direct method writes

$$\hat{u}_B = (1-\theta)u_{-1} + \theta u_0 = u_B + \sum_{k=1}^{\infty} \left[(1-\theta)(-\theta)^k + \theta(1-\theta)^k \right] \frac{h^k}{k!} \frac{\partial^k u}{\partial x^k}, \qquad \text{if } 0 \le \theta \le 1, \tag{41}$$

and the linear method writes



(a) Shifted direct method (b) Shifted linear method

Fig. 14. Coefficients of the Taylor's series of the truncation error.

$$\begin{aligned}\hat{u}_B &= (1-2\theta)u_{-1} + 2\theta u_0 \\ &= u_B + \sum_{k=1}^{\infty}\left[(1-\theta)(-\theta)^k + \theta(1-\theta)^k\right]\frac{h^k}{k!}\frac{\partial^k u}{\partial x^k}, \qquad \text{if } 0 \le \theta \le \frac{1}{2}, \end{aligned} \tag{42a}$$

$$\begin{aligned}\hat{u}_B &= (2-2\theta)u_0 + (2\theta-1)u_{+1} \\ &= u_B + \sum_{k=1}^{\infty}\left[(\theta-\tfrac{1}{2})(2-\theta)^k + (1-\theta)(1-\theta)^k + \tfrac{1}{2}(-\theta)^k\right]\frac{h^k}{k!}\frac{\partial^k u}{\partial x^k}, \qquad \text{if } \frac{1}{2} \le \theta \le 1. \end{aligned} \tag{42b}$$

Notice that the compactness of the linear method is 1 when $0 \le \theta \le \frac{1}{2}$ and 2 when $\frac{1}{2} \le \theta \le 1$. Coefficients of the first eight terms of the series are shown in Fig. 14a for the direct method, and in Fig. 14b for the linear method. Coefficients are identical when $0 \le \theta \le \frac{1}{2}$, whereas coefficients of the linear method have much higher values when $\frac{1}{2} \le \theta \le 1$. We therefore can expect higher error values from the linear method when the compactness is 2, which can reasonably explain the results of the two dimensional computation.

5.2. Neumann immersed boundary condition for the circular interface

Analytical solution. Another solution of the Poisson equation is given by the solution and source fields

$$u(x,y) = (1+x)^3, \qquad f(x,y) = -6(1+x), \qquad \forall (x,y) \in \Omega_i, \tag{43}$$

provided consistent computational domain boundary conditions

$$u(-1,y) = 0, \qquad u(+1,y) = 8, \qquad \forall y \in [-1,+1], \tag{44a}$$

$$\frac{\partial u}{\partial y}(x,-1) = 0, \qquad \frac{\partial u}{\partial y}(x,+1) = 0, \qquad \forall x \in [-1,+1], \tag{44b}$$

$$\partial_{\boldsymbol{n}} u(x,y) = 6x(1+x)^2, \qquad \forall (x,y) \in \Gamma. \tag{44c}$$

The truncation terms for this particular solution are

$$F_{i,j} - \frac{U_{i-1,j} - U_{i,j}}{h_x^2} - \frac{U_{i+1,j} - U_{i,j}}{h_x^2} - \frac{U_{i,j-1} - U_{i,j}}{h_y^2} - \frac{U_{i,j+1} - U_{i,j}}{h_y^2} = 0, \quad \forall (i,j) \in C_i, \tag{45}$$

$$\begin{aligned} &0 - \frac{U_{0,j} + U_{1,j}}{2} = 0, \quad 8 - \frac{U_{m,j} + U_{m+1,j}}{2} = -\tfrac{3}{2}h_x^2, \quad \forall j \in [\![1,n]\!], \\ &0 - \frac{U_{i,1} - U_{i,0}}{h_y} = 0, \quad 0 - \frac{U_{i,m+1} - U_{i,m}}{h_y} = 0, \qquad \forall i \in [\![1,m]\!]. \end{aligned} \tag{46}$$

Again, only the Dirichlet boundary conditions at $\partial\Omega_l$ and $\partial\Omega_r$ will be numerically corrected.

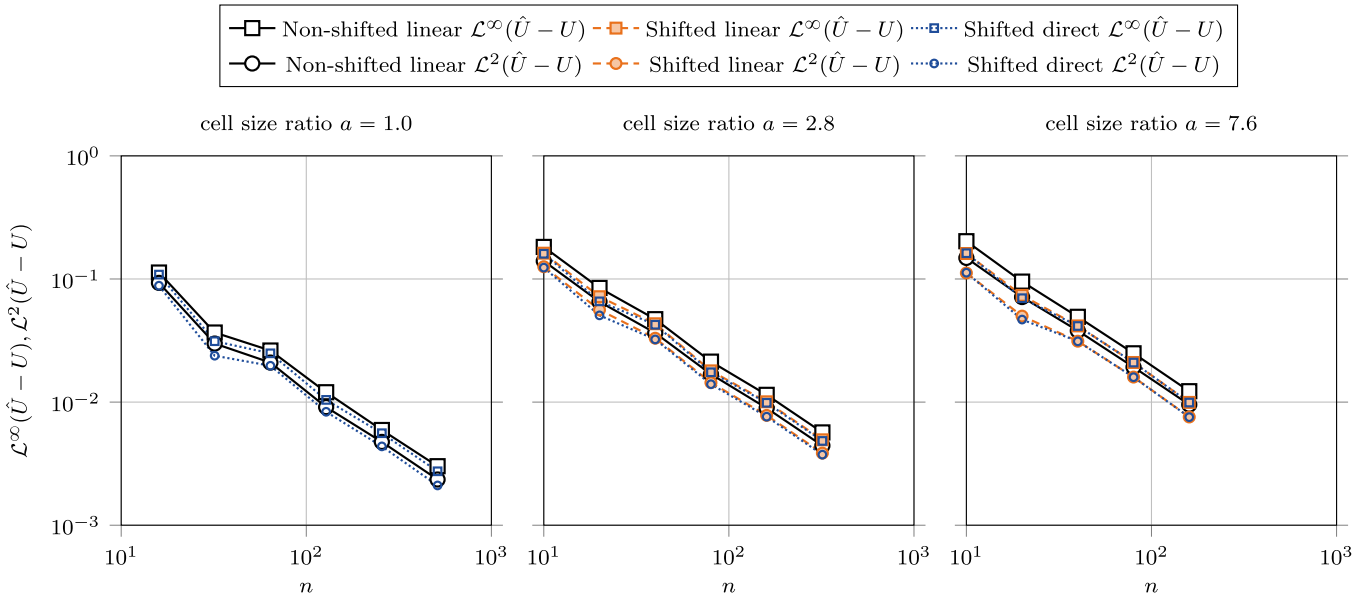


Fig. 15. Norms of the solution error $\hat{U} - U$ of the Poisson problem for Neumann immersed boundary condition on the circular interface Sect. 5.2. Each plot represents a mesh series. Solutions have been computed with the non-shifted linear, shifted linear, and shifted direct methods, except for the shifted linear method on the mesh series $a = 1.0$ as it is equivalent to the non-shifted linear method. The x-axis represents $1/h = 1/\max\{h_x, h_y\} = \min\{n_x, n_y\}$, which is always equal to n in this section.

Results with second-order interpolation. Numerical solutions $\hat{U}$ are computed using the methods described in Sect. 5.1. Second-order interpolations ($p = 2$) are used to build immersed boundary conditions. Fig. 15 shows $\mathcal{L}^2$ and $\mathcal{L}^\infty$ norms of the error field $\hat{U} - U$. As expected, a first-order limiting behavior is obtained with all methods. As with Dirichlet boundary conditions, the figure also shows that, for any given grid, the shifted linear methods SL and SD have less errors than the non-shifted linear method NL. The two shifted methods SL and SD are equivalent in terms of both $\mathcal{L}^2$ and $\mathcal{L}^\infty$ norms.

Results with third-order interpolation. The same computations are performed again, but third order interpolations ($p = 3$) are used to build immersed boundary conditions. Fig. 16 shows $\mathcal{L}^2$ and $\mathcal{L}^\infty$ norms of the error field $\hat{U} - U$. As expected, a second-order limiting behavior is obtained with all methods. As with second-order interpolations, the figure also shows that, for any given grid, the shifted linear methods SL and SD have less errors than the non-shifted linear method NL. The two shifted methods SL and SD are equivalent in terms of both $\mathcal{L}^2$ and $\mathcal{L}^\infty$ norms.

5.3. Dirichlet immersed boundary condition for the flower-shaped interface

The parabolic analytical solution Eq. (37) with Dirichlet boundary conditions is also applied here. Numerical solutions $\hat{U}$ are computed using the shifted linear (SL) and the shifted direct (SD) methods for all series. As the anisotropy increases, the low resolved grids become less and less adapted to the immersed boundary, and the hollow case problem discussed in Sect. 2.3 starts to occur. This is shown in Fig. 17 where a row of four inner nodes are surrounded by ghost cells only, except on the left side.

To complete this convergence study, we used another grid series with $a = 4.7$, and we increase the resolution for $a = 7.6$ to catch the limiting second-order convergence region. Fig. 18 shows $\mathcal{L}^2$ and $\mathcal{L}^\infty$ norms of the error field $\hat{U} - U$. As expected, a second-order limiting behavior is obtained with all methods. Here again, error levels of the SD method are less or equal than error levels of the SL method. For sufficiently high resolution, a convergence is recovered for the grid series $a = 7.6$.

5.4. Solver comparison

Simulations presented in Sect. 5.3 and Sect 5.2 have been used to compare the performance of different linear system solvers. As the stencil size is 1 for the direct method, we use the GMRES solver with the SMG preconditioner from the *hypre* library (GMRES+SMG), and as the stencil size of the linear method is 2, we use the GMRES solver with the BoomerAMG preconditioner from the same library (GMRES+BoomerAMG). The allowed residual for all solvers is 10^{-10}. Also, the *mumps* solver has been used in both cases, for which we do not count time take by the symbolic factorization, which is quite long. Computations have been done in parallel using 16 processes on Intel® Xeon® CPU E5-4640 0 at 2.40 GHz, and solver details are given in Appendix B.

If we give a look to the flower test case with Dirichlet boundary conditions in Table 3, between the two iterative solvers, GMRES+BoomerAMG is faster than GMRES+SMG at low resolutions, but the former slows much more rapidly as the

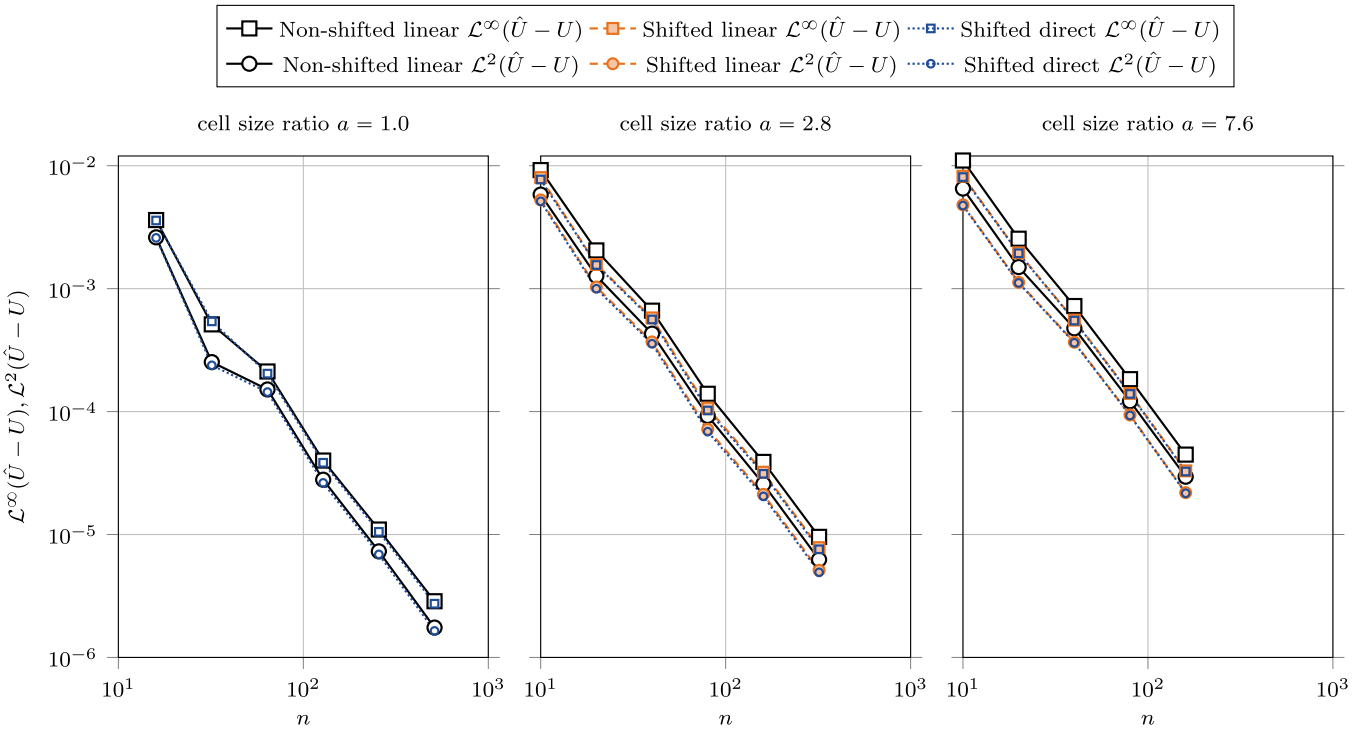


Fig. 16. Norms $\mathcal{L}^2$ and $\mathcal{L}^\infty$ of the solution error $\hat{U}-U$ for the Poisson problem with Neumann immersed boundary conditions and third-order interpolation, as presented in Sect. 5.2. The non-shifted linear, shifted linear, and shifted direct methods are computed in each of the three mesh series. Although, the shifted linear method is not shown for the mesh series $a = 1$, as it is equivalent to the non-shifted linear method. The x-axis represents $1/h = 1/\max\{h_x, h_y\} = \min\{n_x, n_y\}$, which is always equal to n in this section.

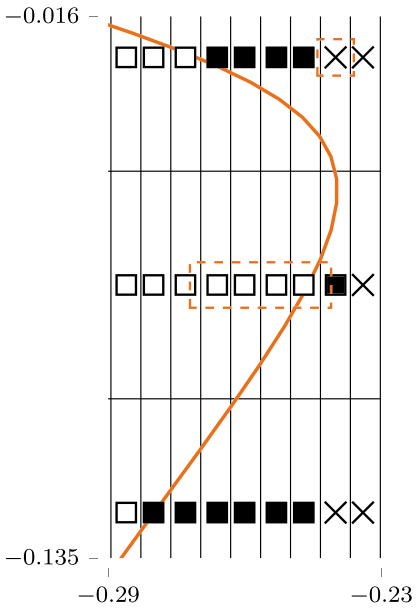


Fig. 17. Close view of node types near a hollow part of the Flower-shaped immersed boundary with a gird of ratio $a = 7.6$. Outlined by a dashed box, a row of four inner cells are surrounded by ghost cells except on the left. Also outlined by a dashed box, an outer cell whose value is used in interpolation due to the hollow case problem. Large error is expected in this region.

grid size increases. It confirms the interest we have to get the most compact scheme. It is also another advantage of the direct method which seems to be more precise than the linear method. Note that for the circular interface with Neumann immersed boundary conditions in Table 4, for the finest mesh and $a = 4.7$, both GMRES+SMG and GMRES+BoomerAMG fail or converge very slowly; instead we used *mumps*. The direct *mumps* solver is slower than their iterative counterparts, yet competitive; it is also less affected by the anisotropy a and is even faster than GMRES+SMG in one case.

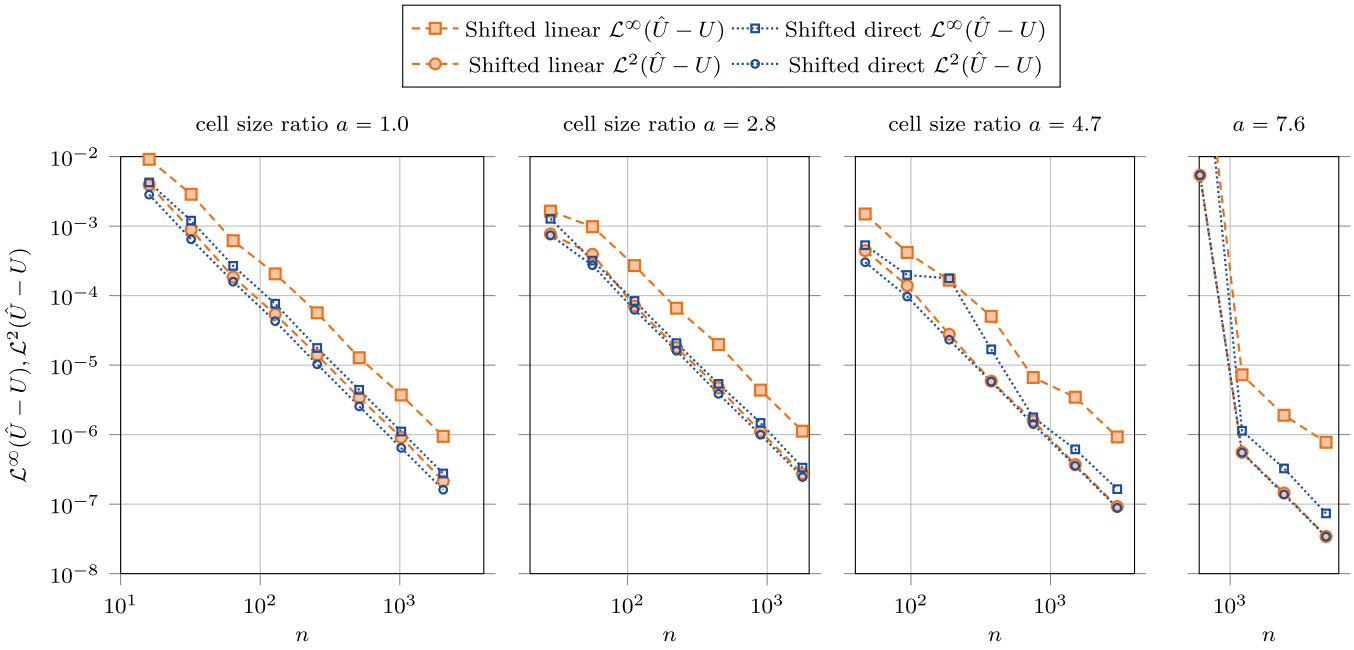


Fig. 18. Norms $\mathcal{L}^2$ and $\mathcal{L}^\infty$ of the solution error $\hat{U} - U$ for the Poisson problem with Flower-shaped immersed boundary. The x-axis represents $1/h = 1/\max\{h_x, h_y\} = \min\{n_x, n_y\}$, which is always equal to n in this section.

Table 3
Comparison of the performance of linear system solvers to solve the Poisson problem with a Flower-shaped immersed boundary and Dirichlet boundary conditions Sect. 5.3 in terms of GMRES iterations (*Iter.* columns) and CPU time (*Time* columns). The tolerance of the GMRES solver is 10^{-10}. *Mumps* symbolic factorization is not taken into account in the CPU time. Computations have been done in parallel with 16 processes on Intel® Xeon® CPU E5-4640 0 at 2.40 GHz.

a	Grid	Direct method			Linear method		
		GMRES+SMG		MUMPS	GMRES+BoomerAMG		MUMPS
		Iter.	Time (s)	Time (s)	Iter.	Time (s)	Time (s)
1	160×160	15	0.06	0.05	18	0.04	0.09
	320×320	19	0.13	0.14	19	0.12	0.31
	640×640	23	0.35	0.56	21	0.65	1.32
	1280×1280	24	1.35	2.71	19	4.01	9.93
7.6	608×80	18	0.08	0.07	23	0.07	0.12
	1216×160	17	0.16	0.22	25	0.34	0.53
	2432×320	21	0.54	0.88	28	1.7	3.5
	4864×640	29	6.25	4.2	31	24.6	19.7

Table 4
Comparison of the performance of linear system solvers to solve the Poisson problem with a circular immersed boundary and Neumann boundary conditions Sect. 5.2 in terms of GMRES iterations (*Iter.* columns) and CPU time (*Time* columns). The tolerance of the GMRES solver is 10^{-10}. *Mumps* symbolic factorization is not taken into account in the CPU time. Computations have been done in parallel with 16 processes on Intel® Xeon® CPU E5-4640 0 at 2.40 GHz.

a	Grid	Direct method			Linear method		
		GMRES+SMG		MUMPS	GMRES+BoomerAMG		MUMPS
		Iter.	Time (s)	Time (s)	Iter.	Time (s)	Time (s)
2.8	448×160	39	0.06	0.11	43	0.1	0.2
	896×320	46	0.30	0.38	61	0.9	0.75
	1792×640	41	1.42	1.54	66	4.38	5.10
	3584×1280	78	14	8.7	126	38	37
4.7	376×80	62	0.05	0.06	246	0.23	0.06
	752×160	138	0.31	0.12	> 500	n/a	0.31
	1504×320	> 500	n/a	0.59	> 500	n/a	1.90
	3008×640	> 500	n/a	3.28	> 500	n/a	8.8

6. Immersed boundary method with the incompressible Navier–Stokes problem

This section applies the shifted immersed boundary method to the incompressible Navier–Stokes problem with uniform density and viscosity. A pressure-correction method on a staggered grid is employed. It is shown here that just applying the Ghost-Fluid Finite-Difference to the sub-problems is not enough, and, due to the nonlinear term, an additional extrapolation is necessary.

We consider the *velocity field* $\boldsymbol{u} = (u, v)^\intercal : \Omega_i \times [0, T] \to \mathbb{R}^2$ and the *kinematic pressure* $p : \Omega_i \times [0, T] \to \mathbb{R}$, both restricted to the inner domain, solution of the incompressible Navier–Stokes with uniform density, uniform viscosity, and *source field* $\boldsymbol{f} : \Omega_i \times [0, T] \to \mathbb{R}^2$:

$$\partial_t \boldsymbol{u} + \nabla\cdot(\boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \nu \Delta \boldsymbol{u} + \boldsymbol{f}\,, \qquad \text{in } \Omega_i\,, \tag{47a}$$

$$\nabla\cdot \boldsymbol{u} = 0\,, \qquad \text{in } \Omega_i\,, \tag{47b}$$

$$\boldsymbol{u} = \boldsymbol{D}\,, \qquad \text{on } \partial\Omega_D \cup \Gamma_D\,, \tag{47c}$$

$$\nabla \boldsymbol{u} \cdot \boldsymbol{n} = \boldsymbol{N}\,, \qquad \text{on } \partial\Omega_N \cup \Gamma_N\,, \tag{47d}$$

where ν is the *kinematic viscosity*, and $\boldsymbol{D}$ and $\boldsymbol{N}$ are known functions.

6.1. Time discretization and velocity–pressure coupling

The timeline $[0, T]$ is divided in constant *time steps* h_t. Equation (47) is applied at t^{n+1}, the time derivative $\partial_t \boldsymbol{u}$ is developed backwards, and the nonlinear term is linearized:

$$\frac{1}{h_t}\left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n\right) + \nabla\cdot(\boldsymbol{u}^{n+1} \otimes \boldsymbol{u}^n) = -\nabla p^{n+1} + \nu \Delta \boldsymbol{u}^{n+1} + \boldsymbol{f} + \mathcal{O}(h_t)\,, \qquad \text{in } \Omega_i\,, \tag{48a}$$

$$\nabla\cdot \boldsymbol{u}^{n+1} = 0\,, \qquad \text{in } \Omega_i\,, \tag{48b}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{D}\,, \qquad \text{on } \partial\Omega_D \cup \Gamma_D\,, \tag{48c}$$

$$\nabla \boldsymbol{u}^{n+1} \cdot \boldsymbol{n} = \boldsymbol{N}\,, \qquad \text{on } \partial\Omega_N \cup \Gamma_N\,. \tag{48d}$$

In Eq. (48a), errors introduced by the terms $\frac{1}{h_t}\left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n\right)$ and $\nabla\cdot(\boldsymbol{u}^{n+1} \otimes \boldsymbol{u}^n)$ are collected in $\mathcal{O}(h_t)$; the pressure gradient term ∇p^{n+1} remains undefined. To solve the velocity–pressure coupling, we use the rotational incremental pressure-correction scheme [18,19], in which Eq. (48) is divided in two subproblems: a prediction step that does not satisfy the solenoidal constraint Eq. (48b), and a correction step. The prediction step consists in solving the advection–diffusion problem

$$\frac{1}{h_t}\left(\boldsymbol{u}^{*} - \boldsymbol{u}^n\right) + \nabla\cdot(\boldsymbol{u}^{*} \otimes \boldsymbol{u}^n) = -\nabla p^{n} + \nu \Delta \boldsymbol{u}^{*} + \boldsymbol{f} + \mathcal{O}(h_t)\,, \qquad \text{in } \Omega_i\,, \tag{49a}$$

$$\boldsymbol{u}^{*} = \boldsymbol{D}\,, \qquad \text{on } \partial\Omega_D \cup \Gamma_D\,, \tag{49b}$$

$$\nabla \boldsymbol{u}^{*} \cdot \boldsymbol{n} = \boldsymbol{N}\,, \qquad \text{on } \partial\Omega_N \cup \Gamma_N\,, \tag{49c}$$

in which the pressure p^{n+1} has been replaced by p^n to remove the velocity–pressure coupling. The solution $\boldsymbol{u}^*$ of Eq. (49) is a prediction of $\boldsymbol{u}^{n+1}$, for which $\nabla\cdot\boldsymbol{u}^*$ is not equal to zero, and $\boldsymbol{u}^{n+1}$ can be expressed from the difference between Eq. (48) and Eq. (49) multiplied by h_t

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{*} - h_t \nabla \phi + \mathcal{O}(h_t)\,, \qquad \text{in } \Omega_i\,, \tag{50a}$$

$$\phi = p^{n+1} - p^n + \nu\, \nabla\cdot \boldsymbol{u}^{*}\,, \qquad \text{in } \Omega_i\,, \tag{50b}$$

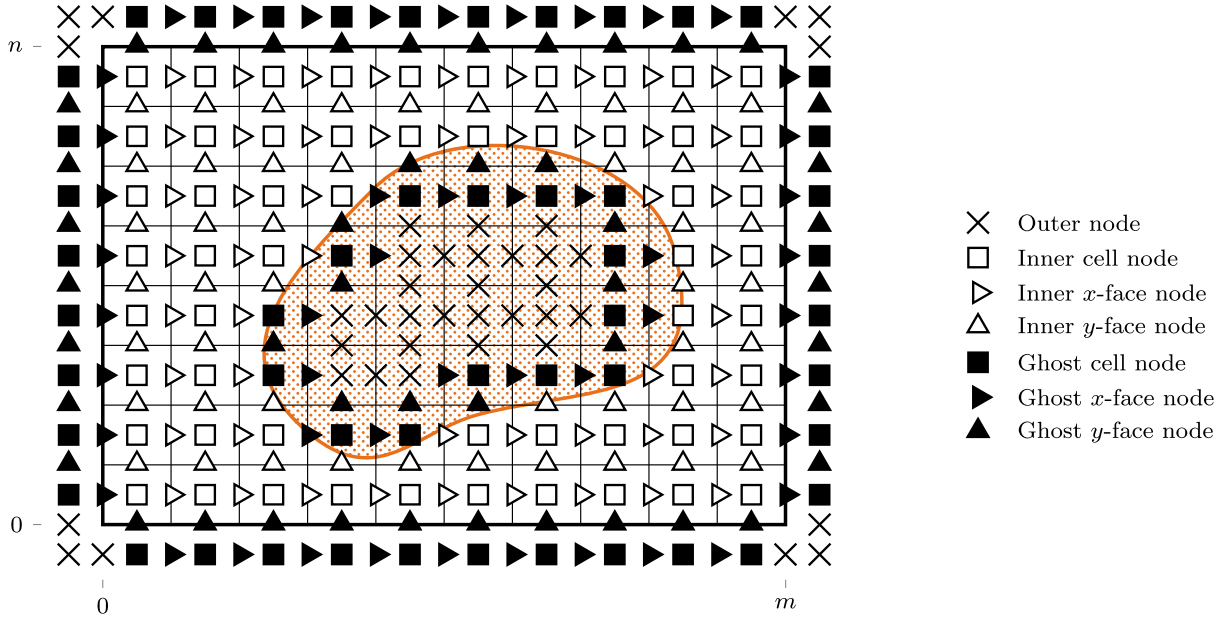
where the decomposition of the Laplace operator $\Delta = \nabla(\nabla\cdot) - \nabla\times\nabla\times$ leads to the term $\nu\,\nabla\cdot\boldsymbol{u}^*$ in Eq. (50b), and the seemingly missing terms $-h_t\,\nabla\cdot\left((\boldsymbol{u}^{n+1} - \boldsymbol{u}^{*}) \otimes \boldsymbol{u}^n\right) + \nu h_t\,\nabla\times\nabla\times(\boldsymbol{u}^{n+1} - \boldsymbol{u}^{*})$ are in fact collected in $\mathcal{O}(h_t)$. In Equation (50) the *pressure increment* ϕ is obtained by solving the Poisson problem

$$\nabla\cdot(h_t \nabla \phi) = \nabla\cdot \boldsymbol{u}^{*} + \mathcal{O}(h_t)\,, \qquad \text{in } \Omega_i\,, \tag{51a}$$

$$\nabla \phi \cdot \boldsymbol{n} = 0\,, \qquad \text{on } \partial\Omega_D \cup \Gamma_D\,, \tag{51b}$$

$$\phi = 0\,, \qquad \text{on } \partial\Omega_N \cup \Gamma_N\,. \tag{51c}$$

Equations (49)–(51) form the time-discretized Navier–Stokes equations used in this paper.



(a) Sketch of a Cartesian grid with node types that can be used to discretize Fig. 1a.

Fig. 19. Definitions and notations of the computational domain, the immersed boundary, the mesh, and node types.

6.2. Spatial discretization

A staggered discretization of fields is used: to cell nodes, we consider x-face nodes $\boldsymbol{X}_{i+\frac{1}{2},j}$, $\forall (i,j) \in [\![0,m]\!] \times [\![0,n+1]\!]$, and y-face nodes $\boldsymbol{X}_{i,j+\frac{1}{2}}$, $\forall (i,j) \in [\![0,m+1]\!] \times [\![0,n]\!]$. Vector fields are discretized on face nodes: $\boldsymbol{U} = (U,V)^\intercal$ where $U_{i+\frac{1}{2},j} = u(\boldsymbol{X}_{i+\frac{1}{2},j})$ and $V_{i,j+\frac{1}{2}} = v(\boldsymbol{X}_{i,j+\frac{1}{2}})$, and $\boldsymbol{F} = (F,G)^\intercal$ where $F_{i+\frac{1}{2},j} = f(\boldsymbol{X}_{i+\frac{1}{2},j})$ and $G_{i,j+\frac{1}{2}} = g(\boldsymbol{X}_{i,j+\frac{1}{2}})$. Cell types are assigned using the method Eq. (2) defined in Sect. 2.1. It is applied to cell nodes, x-face nodes, and y-face nodes independently, as shown in Fig. 19. This way ensures that the discretized operators defined below are well-defined. Namely, two cell nodes that surround an inner face node are either inner or ghost, two face nodes that surround an inner cell node are either inner or ghost, and y-face nodes (resp. x-face nodes) that surround an x-face node (resp. y-face node) are either inner or ghost. The set of inner, ghost, outer x-face indices are C_i^x, C_g^x, C_o^x, respectively, and the set of inner, ghost, outer y-face indices are C_i^y, C_g^y, C_o^y.

Each term of Eq. (49)–(51) discretizes with second-order centered schemes as described in Appendix A; this results in

$$\left(\frac{1}{h_t}\boldsymbol{I}_i + \boldsymbol{A}(\boldsymbol{U}^n) - \nu\boldsymbol{L} + \boldsymbol{E}\right)\boldsymbol{U}^* = \frac{1}{h_t}\boldsymbol{U}^n - \boldsymbol{G}P^n + \boldsymbol{F} + \boldsymbol{B} + \mathcal{O}(h_t) + \mathcal{O}(h^2) + \mathcal{O}(h^p)\,, \qquad \text{in } C_{ig}^{xy}\,, \tag{52a}$$

$$h_t(L+E)\Phi = D\boldsymbol{U}^* + B + \mathcal{O}(h_t) + \mathcal{O}(h^2) + \mathcal{O}(h^{q-1})\,, \qquad \text{in } C_{ig}\,, \tag{52b}$$

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^* - h_t\boldsymbol{G}\Phi + \mathcal{O}(h_t) + \mathcal{O}(h^2)\,. \qquad \text{in } C_i^{xy}\,, \tag{52c}$$

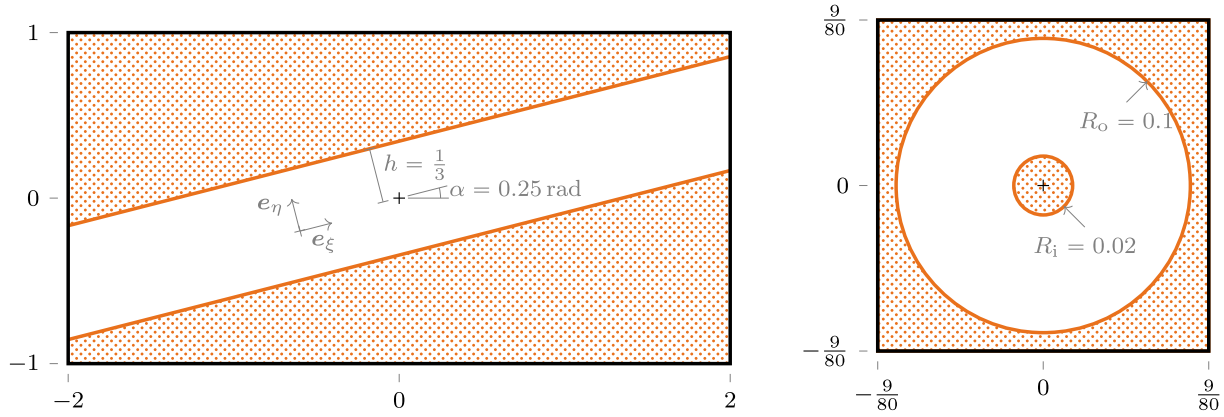
$$P^{n+1} = P^n + \Phi - \nu D\boldsymbol{U}^* + \mathcal{O}(h^2)\,, \qquad \text{in } C_i\,, \tag{52d}$$

where $C_{ig}^{xy} = C_{ig}^x \cup C_{ig}^y$, and $\boldsymbol{I}_i$ is the identity matrix over the inner cells and the null matrix elsewhere. As in Sect. 2.1, we closed the two first equations using domain and immersed boundary conditions: $\boldsymbol{E}$ and $\boldsymbol{B} + \mathcal{O}(h^p)$ in Eq. (52a) and E and $B + \mathcal{O}(h^{q-1})$ in Eq. (52b). We recall that $(\boldsymbol{E}\boldsymbol{U}^*)_{i,j} = \boldsymbol{B}_{i,j} = 0$ over C_i^{xy}, $(E\Phi)_{i,j} = B_{i,j} = 0$ over C_i, and the other terms evaluate to zero over C_g^{xy} and C_g accordingly. Although it is not trivial that terms $\boldsymbol{A}(\boldsymbol{U}^n)$, $D\boldsymbol{U}^*$, $\boldsymbol{G}P^n$, and $\boldsymbol{G}\Phi$ are well defined over C_i^{xy}, so let's have a closer look to these terms.

According to interpolations and finite-differences involved in Eq. (A.2), $\boldsymbol{A}(\boldsymbol{U}^n)$ will be defined over C_i^{xy}, if $\boldsymbol{U}^n$ is defined over C_{ig}^{xy}. Therefore the whole iterative process Eq. (52) must output a $\boldsymbol{U}^{n+1}$ defined over C_{ig}^{xy}. To this end, the right-hand side of Eq. (52c) must be extended over the ghost nodes: $\boldsymbol{U}^*$ is already extended since it is a solution of Eq. (52a), although $\boldsymbol{G}\Phi$ cannot be defined over all ghost nodes (because is would require to define Φ over some outer nodes). An extrapolation of the velocity field must then be added:

$$(\boldsymbol{I}_i + \boldsymbol{E})\tilde{\boldsymbol{U}}^{n+1} = \boldsymbol{U}^{n+1} + \boldsymbol{B} + \mathcal{O}(h^p)\,, \qquad \text{in } C_{ig}^{xy}\,, \tag{53}$$

where $\tilde{\boldsymbol{U}}^{n+1}$ is the new output of Eq. (52).



(a) Poiseuille flow. (b) Couette flow.

Fig. 20. Computational domain and immersed boundary setups for the Poiseuille flow and the Couette flow. A thick black line shows the computational domain, a thick colored line shows the immersed boundary and colored dots shades the outer domain.

According to Eq. (A.3b), $\boldsymbol{G}P^n$ will be defined over C_i^{xy}, if P^n is defined over C_i plus ghost nodes next to an inner face node. Therefore the whole iterative process Eq. (52) must output a P^{n+1} defined over C_i plus ghost nodes next to an inner face node. To this end, the right-hand side of Eq. (52d) must be extended over the ghost nodes next to an inner face node: Φ is already defined over C_{ig} since it is a solution of Eq. (52b), $-\nu D\boldsymbol{U}^*$ is also defined over ghost nodes next to an inner face node since the opposite face node is necessarily a ghost node where $\boldsymbol{U}^*$ is defined and a finite-difference can be applied. As a consequence, no extrapolation is necessary for P^{n+1}.

To summarize, the fully discretized Navier–Stokes equations with immersed boundaries are

$$\left(\frac{1}{h_t}\boldsymbol{I}_i + \boldsymbol{A}(\boldsymbol{U}^n) - \nu\boldsymbol{L} + \boldsymbol{E}\right)\boldsymbol{U}^* = \frac{1}{h_t}\boldsymbol{U}^n - \boldsymbol{G}P^n + \boldsymbol{F} + \boldsymbol{B} + \mathcal{O}(h_t) + \mathcal{O}(h^2) + \mathcal{O}(h^p)\,, \qquad \text{in } C_{ig}^{xy}\,, \tag{54a}$$

$$h_t(L+E)\Phi = D\boldsymbol{U}^* + B + \mathcal{O}(h_t) + \mathcal{O}(h^2) + \mathcal{O}(h^{q-1})\,, \qquad \text{in } C_{ig}\,, \tag{54b}$$

$$(\boldsymbol{I}_i + \boldsymbol{E})\boldsymbol{U}^{n+1} = \boldsymbol{U}^* - h_t\boldsymbol{G}\Phi + \boldsymbol{B} + \mathcal{O}(h_t) + \mathcal{O}(h^2) + \mathcal{O}(h^p)\,, \qquad \text{in } C_{ig}^{xy}\,, \tag{54c}$$

$$P^{n+1} = P^n + \Phi - \nu D\boldsymbol{U}^* + \mathcal{O}(h^2)\,, \qquad \text{in } C_i\,, \tag{54d}$$

with the precaution that $\boldsymbol{U}^0$ and P^0 have also been extrapolated over the ghost nodes. Immersed boundary conditions are applied three times: in Eq. (54a) and Eq. (54b), and an immersed boundary extrapolation is applied Eq. (54c).

7. Numerical simulations with the incompressible Navier–Stokes problem

This section presents simulations of four cases of the Navier–Stokes problem, and compares results obtained with both shifted direct and shifted linear methods. The size of the discretization stencil is the same for prediction and correction steps: $p = q = 2$, which equals to one for the direct method and two for the linear method.

Linear system solvers. The same solvers are used than in Sect. 5, except for the prediction step for which a Jacobi preconditioner is enough. Stationarity tolerance is set to 10^{-8}. Details of the linear system solvers are given in Appendix B.

Diagnostics. For the two first cases, an analytical solution is available. Velocity and pressure error fields, $\hat{\boldsymbol{U}} - \boldsymbol{U}$ and $\hat{P} - P$, are computed at inner nodes, and their $\mathcal{L}^2$ and $\mathcal{L}^\infty$ norms are computed as in Eq. (34). Diagnostics for the two last cases will be presented in Sect. 7.3.

7.1. Poiseuille flow

Computational domain and grid. This computation takes place in the Cartesian domain $\Omega = [-2, +2] \times [-1, +1]$. The immersed boundary is a tilted channel of half-height $h = \frac{1}{3}$, and the angle between the x-axis and the axis of the channel is $\alpha = 0.25$ rad, as shown in Fig. 20a. Dirichlet boundary conditions are used at all boundaries, taking values of the analytical solution presented below. The same number of cells is used in both directions, so that the cell size ratio is always $a = 2$.

Analytical solution. Using the coordinate system $(\boldsymbol{e}_\xi, \boldsymbol{e}_\eta)$, aligned with the channel (see Fig. 20a), the classical solution of the Poiseuille flow is

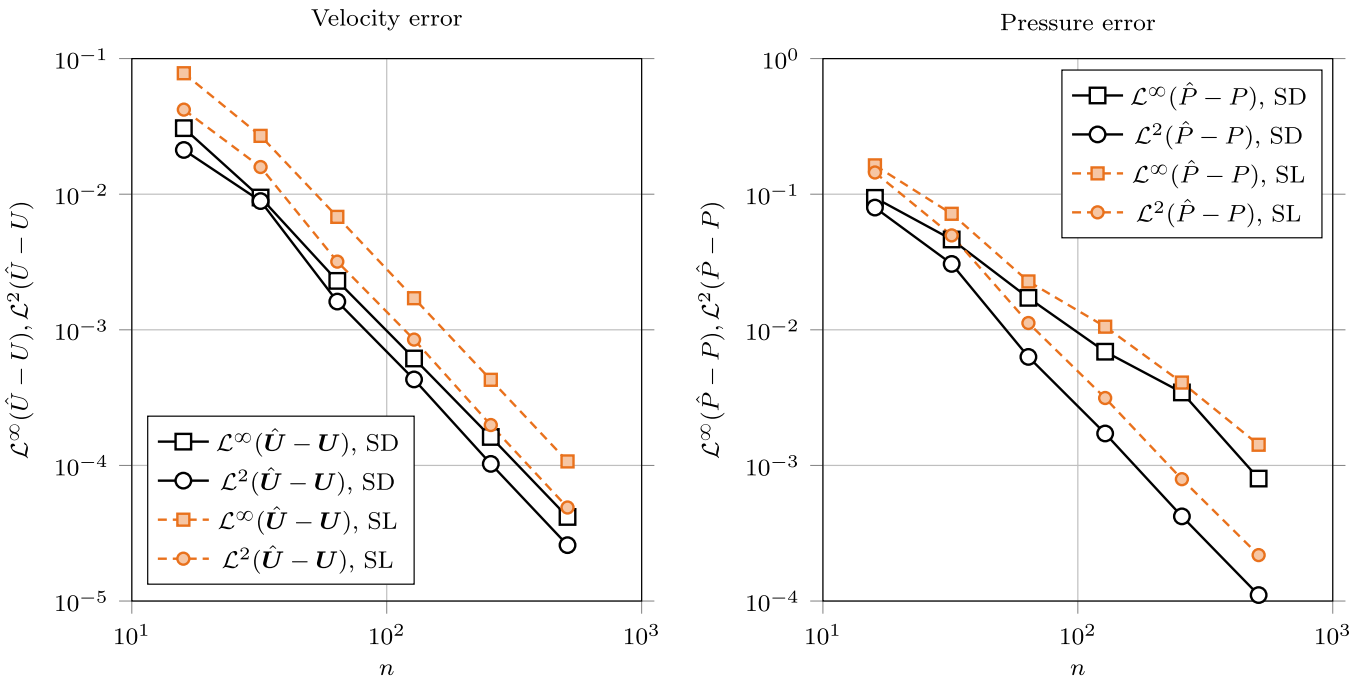


Fig. 21. Norms $\mathcal{L}^2$ and $\mathcal{L}^\infty$ of the velocity error $\hat{\boldsymbol{U}}-\boldsymbol{U}$ and the pressure error $\hat{P}-P$ for the Poiseuille flow using both the shifted direct (SD) and shifted linear (SL) methods. The x-axis represents $1/h = 1/\max\{h_x, h_y\} = \min\{n_x, n_y\}$, which is always equal to n in this section.

$$\boldsymbol{u} = u_\xi \boldsymbol{e}_\xi\,, \qquad u_\xi(\eta) = u_m\left(1-\frac{\eta^2}{h^2}\right), \tag{55a}$$

$$p(\xi) = p_0 + \left(-\frac{\partial p}{\partial \xi}\right)\xi\,, \qquad \left(-\frac{\partial p}{\partial \xi}\right) = \frac{2\nu}{h^2}u_m\,, \tag{55b}$$

where u_m is the maximum velocity and p_0 is the pressure at the origin. The Reynolds number of the flow is $Re = 2hu_m/\nu$. In the following, $p_0 = 0$, $u_m = 1/(2h)$, $\nu = 1/Re$, and $Re = 20$. The average velocity and pressure of the flow are $\langle \boldsymbol{u}\rangle = \frac{2}{3}u_m\boldsymbol{e}_\xi$ and $\langle p\rangle = p_0 = 0$.

Results. Fig. 21 shows convergence results in terms of velocity and pressure errors $\mathcal{L}^\infty(\hat{\boldsymbol{U}}-\boldsymbol{U})$, $\mathcal{L}^2(\hat{\boldsymbol{U}}-\boldsymbol{U})$, $\mathcal{L}^\infty(\hat{P}-P)$, and $\mathcal{L}^2(\hat{P}-P)$ using shifted direct and shifted linear methods. For both methods, velocity errors and pressure $\mathcal{L}^2$ error show a second-order limiting behavior, while pressure $\mathcal{L}^\infty$ error show a limiting behavior between 1.5 and 2. Error levels of the linear method are roughly twice as large as error levels of the direct method.

As we choose a consistent size of the discretization stencil ($p = q = 2$), the limiting behavior of Eq. (54b) is only of first order. This result is therefore better than expected. We suspect superconvergence due to the one-dimensional linear profile of the pressure field.

7.2. Circular Couette flow

Computational domain and grid. The computational domain is $\Omega = [-\frac{9}{80}, +\frac{9}{80}] \times [-\frac{9}{80}, +\frac{9}{80}]$, and the immersed boundary is two co-axial cylinders shown in Fig. 20b. The radius of the inner and outer cylinders are $R_\mathrm{i} = 0.02$ and $R_\mathrm{o} = 0.1$, respectively. Dirichlet boundary conditions are used at all boundaries, taking values of the analytical solution presented below. The number of cells in the x-axis is four times the number of cells in the y-axis, such that the cell size ratio is always $a = 4$.

Analytical solution. The cylinders are in rotation with angular velocities ω_i and ω_o, respectively. They induce motion to the fluid, and the classical solution of the Couette flow using the polar coordinates $(\boldsymbol{e}_r, \boldsymbol{e}_\theta)$ is

$$\boldsymbol{u} = u_\theta \boldsymbol{e}_\theta\,, \qquad u_\theta(r) = \frac{A}{2}r + \frac{B}{r}\,, \tag{56a}$$

$$p(r) = p_0 + \frac{A^2}{8}r^2 + AB\ln(r) - \frac{B^2}{2r^2}\,, \tag{56b}$$

where p_0 is an integration constant, and constants A and B are deduced from parameters:

$$A = 2\frac{\omega_\mathrm{o}R_\mathrm{o}^2 - \omega_\mathrm{i}R_\mathrm{i}^2}{R_\mathrm{o}^2 - R_\mathrm{i}^2}\,, \qquad B = (\omega_\mathrm{i} - \omega_\mathrm{o})\frac{R_\mathrm{i}^2 R_\mathrm{o}^2}{R_\mathrm{o}^2 - R_\mathrm{i}^2}\,. \tag{57}$$

In the following $p_0 = 0$, $\omega_\mathrm{i} = 0.5$ rad, and $\omega_\mathrm{o} = 0$ rad.

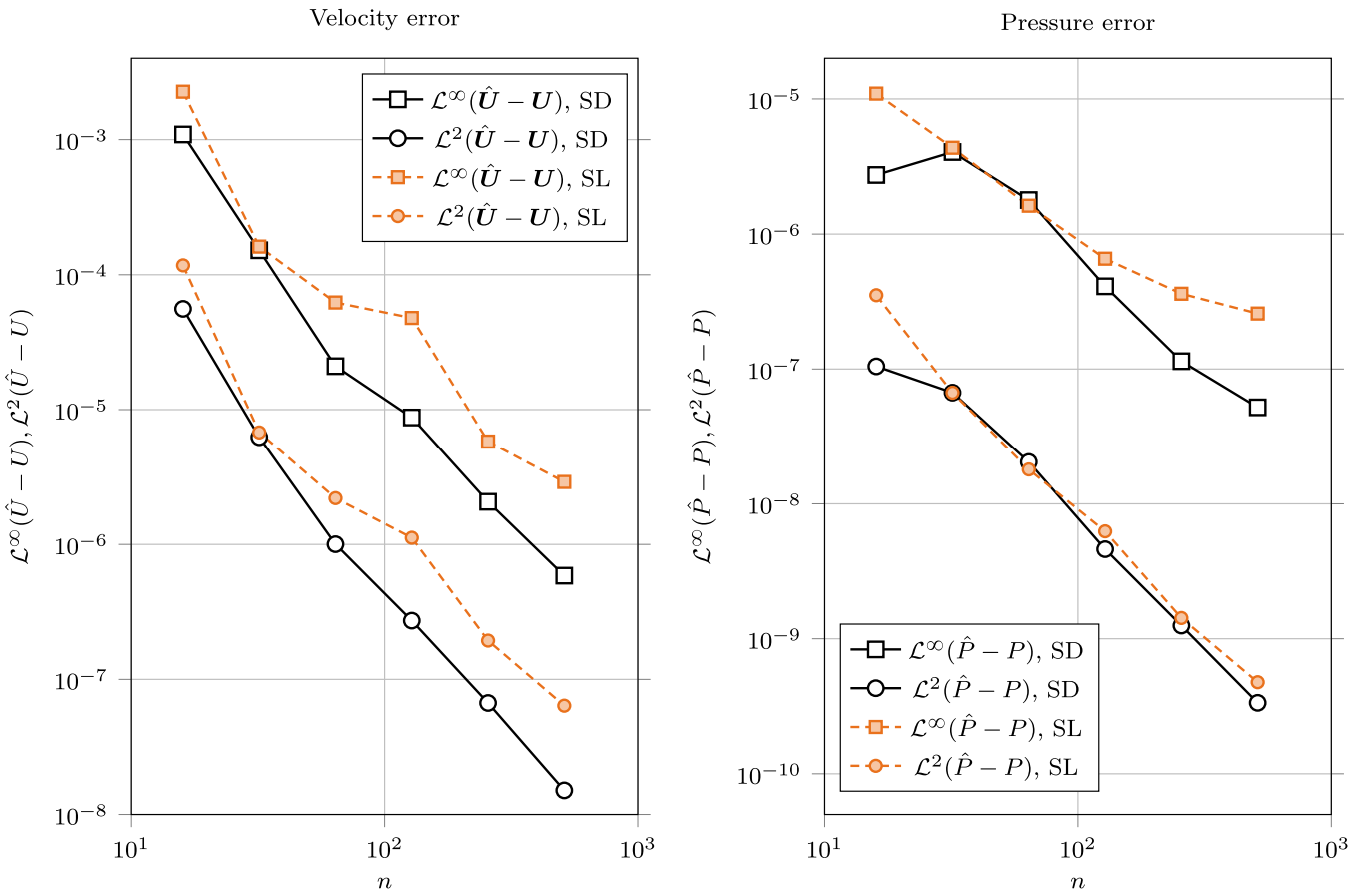


Fig. 22. Norms $\mathcal{L}^2$ and $\mathcal{L}^\infty$ of the velocity error $\hat{\boldsymbol{U}}-\boldsymbol{U}$ and the pressure error $\hat{P}-P$ for the Couette flow using both the shifted direct (SD) and shifted linear (SL) methods. The x-axis represents $1/h = 1/\max\{h_x, h_y\} = \min\{n_x, n_y\}$, which is always equal to n in this section.

Results. Fig. 22 shows convergence results in terms of velocity and pressure errors $\mathcal{L}^\infty(\hat{\boldsymbol{U}}-\boldsymbol{U})$, $\mathcal{L}^2(\hat{\boldsymbol{U}}-\boldsymbol{U})$, $\mathcal{L}^\infty(\hat{P}-P)$, and $\mathcal{L}^2(\hat{P}-P)$. A minimum resolution of 64 cells in the low-resolved axis is required in order to properly represent the inner boundary. Velocity-related indicators show a second-order limiting behavior, whereas $\mathcal{L}^\infty(\hat{P}-P)$ shows only a first-order limiting behavior and $\mathcal{L}^2(\hat{P}-P)$ shows an in-between behavior. As with the Poiseuille flow, error levels of the linear method are roughly twice as large as error levels of the direct method. These results are again consistent with the chosen size of the discretization stencil.

7.3. Flows around circular and elliptic cylinders

The following numerical simulations validate the method on stationary flows around two cylinders of different shapes: circular and elliptic. The former have been extensively studied in the literature, while the latter demonstrates the interest of rectangular cells.

Computational domain and grid. The computational domain represents a tank of the 2-dimensional real space, whose dimensions are $\Omega = [-15, +30] \times [-15, +15]$, and the flow goes from left to right. Domain boundary conditions are: uniform inlet of velocity $(u_\infty, 0)^\intercal$ at the left boundary, slip at top and bottom boundaries, and Neumann at the right boundary. The cylinder is placed at the origin. A regular grid of ratio $a = 3.0$ is placed over a small *domain of interest,* of dimensions $[-1, +3] \times [-1.5, +1.5]$, while exponential grids are placed around. On the y-axis, we used that 512 nodes in the domain of interest and 256 nodes in each exponential region. On the x-axis, we used 1536 nodes in the domain of interest to have $a = 3.0$ and 768 nodes in each stretched region. Fig. 23 shows the computational domain, the grid and the domain of interest.

The circular cylinder has diameter $d = 1$; the elliptic cylinder has major axis $d = 1$ and axis ratio 0.2; thus the characteristic length is d for both cylinders. The major axis of the elliptic cylinder is aligned with the x-axis of the coordinate frame $\Re' = (\boldsymbol{e}_{x'}, \boldsymbol{e}_{y'})$, which is rotated by α with respect to the original coordinate frame $\Re = (\boldsymbol{e}_x, \boldsymbol{e}_y)$ where the x-axis is aligned with the stream. The angle α is also the *angle of incidence;* it is set to $-80°$ for the elliptical cylinder, and can be considered set to $0°$ for the circular cylinder. Fig. 24 shows both cylinders with a description of their dimensions.

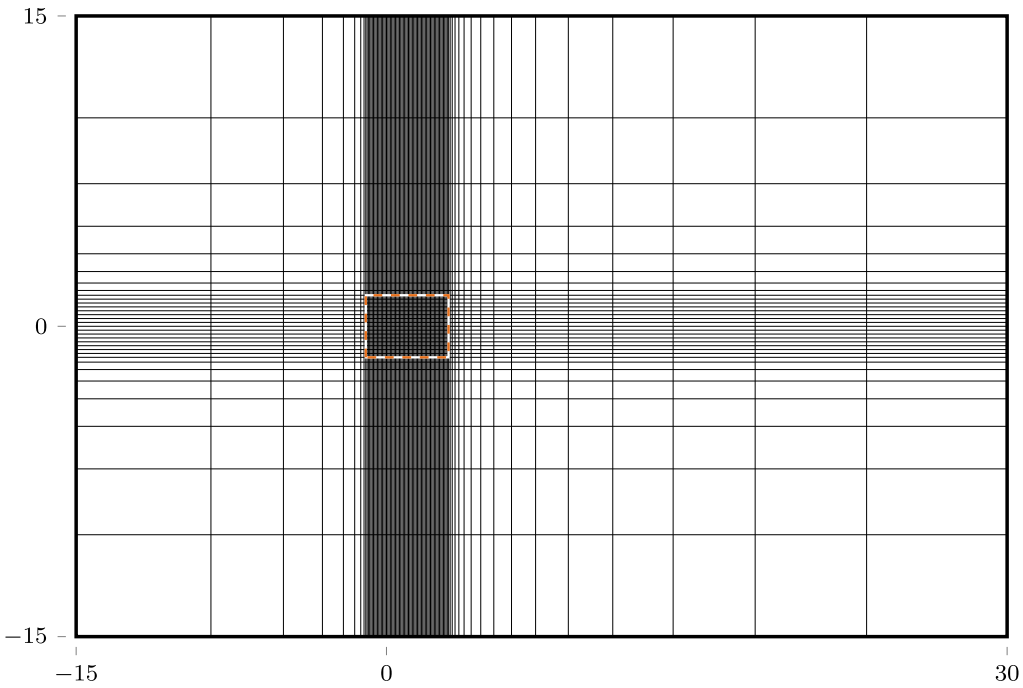
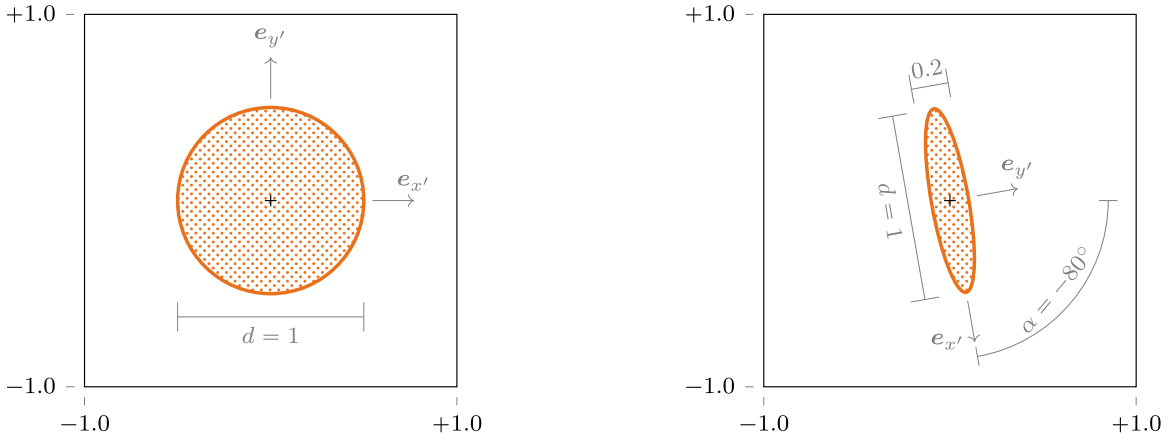


Fig. 23. Computational domain and grid used to simulate the flow around the two cylinders, Sect. 7.3. Only one grid line out of 16 is drawn for the sake of readability. The regular sub-domain region is outlined in dashed colored lines.



(a) Circular cylinder. (b) Elliptic cylinder.

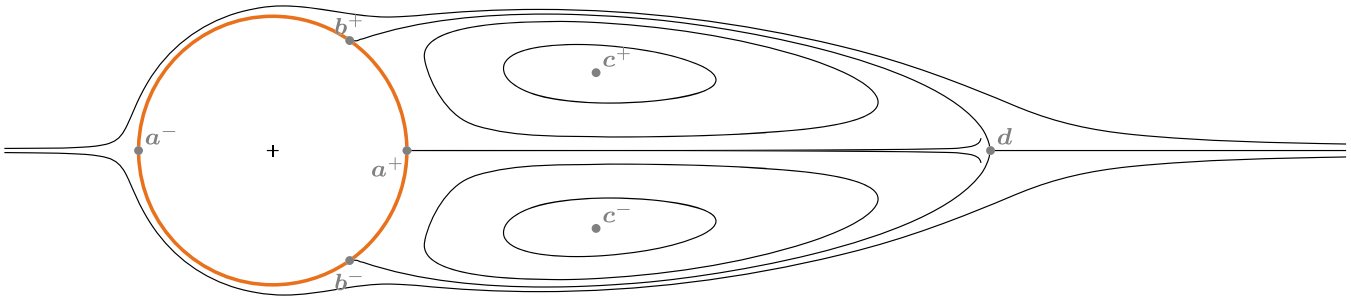
Fig. 24. Dimensions of circular and elliptic cylinders. The boundary is shown by a thick colored line, and the outer domain is shaded by dots. Axes of the cylinder reference frame $\Re' = (\boldsymbol{e}_{x'}, \boldsymbol{e}_{y'})$ are also shown.

We choose the free-stream velocity $u_\infty = 1$ and, and we set the viscosity according to the desired value of the Reynolds number. For the elliptic cylinder, the Reynolds number is based on the long axis, thus $Re = du_\infty/\nu$ both cases. The Reynolds number is set to $Re = 40$ for the circular cylinder, and set to $Re = 20$ for the elliptic cylinder.

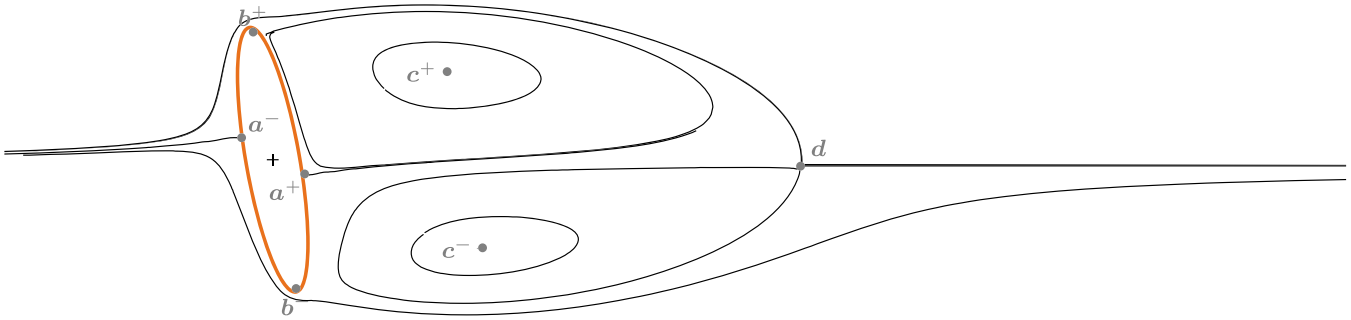
Diagnostics. As no analytical solution exists, we compare our results to experimental and numerical data found in the literature. First, we compare the position of seven different *points of interest:* four separation points on the surface of the obstacle $\boldsymbol{a}^-$, $\boldsymbol{a}^+$, $\boldsymbol{b}^-$, and $\boldsymbol{b}^+$; two vortex cores $\boldsymbol{c}^-$ and $\boldsymbol{c}^+$, and the steady point at the right of the wake vortices $\boldsymbol{d}$. Fig. 25 shows these points for the two cylinders. Cartesian and polar coordinates of a point of interest is noted with the subscript $\boldsymbol{a} = (a_x, a_y)^\intercal$ for the former and $\boldsymbol{a} = (a_r, a_\theta)^\intercal$ for the latter. Next, we compute the drag and lift coefficients

$$C_D = \frac{\mathcal{F}_x}{\frac{1}{2}du_\infty^2}, \qquad C_L = \frac{\mathcal{F}_y}{\frac{1}{2}du_\infty^2}, \tag{58}$$

where $\mathcal{F}_x$ and $\mathcal{F}_y$ are the streamwise and the transverse components of the *volumic drag force* $\boldsymbol{\mathcal{F}}$. This force corresponds to the surface force of the fluid integrated over the cylinder Γ; it decomposes into pressure and viscous components



(a) Circular cylinder ($Re = 40$).



(b) Elliptic cylinder ($Re = 20$).

Fig. 25. Points of interest $\boldsymbol{a}^-$, $\boldsymbol{a}^+$, $\boldsymbol{b}^-$, $\boldsymbol{b}^+$, $\boldsymbol{c}^-$, $\boldsymbol{c}^+$, and $\boldsymbol{d}$ of the steady wake flow. The streamlines are extracted from computations.

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}}_p + \boldsymbol{\mathcal{F}}_\nu = \int_\Gamma -p\boldsymbol{n}\mathrm{d}\ell + \int_\Gamma 2\nu\boldsymbol{\epsilon}\boldsymbol{n}\mathrm{d}\ell\,, \tag{59}$$

where p is the kinematic pressure and $\boldsymbol{\epsilon} = \frac{1}{2}\nu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^\intercal)$ is the strain rate tensor. Drag and lift coefficients can then be decomposed into pressure and viscous components.

$$C_{\mathrm{L}p} = \frac{\mathcal{F}_{py}}{\frac{1}{2}du_\infty^2}\,, \qquad C_{\mathrm{L}\nu} = \frac{\mathcal{F}_{\nu y}}{\frac{1}{2}du_\infty^2}\,, \qquad C_{\mathrm{D}p} = \frac{\mathcal{F}_{px}}{\frac{1}{2}du_\infty^2}\,, \qquad C_{\mathrm{D}\nu} = \frac{\mathcal{F}_{\nu x}}{\frac{1}{2}du_\infty^2}\,. \tag{60a}$$

The numerical evaluation of $\boldsymbol{\mathcal{F}}_p$ and $\boldsymbol{\mathcal{F}}_\nu$ approximates Γ by cell-wise line segments upon which p and $\boldsymbol{\epsilon}$ are considered uniform. Second-order extrapolations are used to compute p and $\boldsymbol{\epsilon}$ on the middle of the line segments, using nodes values from the inner domain only. Finally, we give a closer look at the normalized pressure profile C_p around the cylinder, defined by

$$C_p = \frac{p}{\frac{1}{2}du_\infty^2}\,. \tag{61}$$

Results. Both simulations converge towards a steady state containing two recirculation regions. Both direct and linear methods gives very similar results. Coordinates of the points of interest are reported in Table 5 for the circular cylinder and in Table 6 for the elliptic cylinder. Good agreement with literature data is observed for the circular cylinder; Table 5 also shows data from the literature. Lengths for the elliptic cylinder recirculations are coherent with the streamline visualizations of Yoon et al. [20].

Drag coefficients are reported in Table 7 for the circular cylinder (lift coefficients are below 10^{-8}). The value is in the range of values found in the literature. Drag and Lift coefficients, reported in Table 8 for the elliptic cylinder, are valid for the range of values found in the literature. Most of the drag is due to the pressure component. It is interesting to note that pressure and viscosity have opposite contribution to lift. Finally, profiles of normalized pressure C_p as a function of θ are shown on Fig. 26. The profile for the circular cylinder Fig. 26a follows closely the profiles found in the literature [30–32]. The profile for the elliptic cylinder Fig. 26b has a maximum and minimum pressure than the circular cylinder. Although, very low pressure are seen at the tips of the ellipse ($\theta = 0°, 180°$).

Table 5
Lengths of points of interest for the circular cylinder ($Re = 40$). This table follows the notation used in the literature: $\ell = d_x - a_x^+$, $a = c_x^+ - a_x^+$, $b = c_y^+ - c_y^-$, and θ is the angular coordinate of $\boldsymbol{b}^+$. The coefficient λ represents the size ratio between the cylinder and the tank height (i.e. the height of the computational domain). Both direct and linear method gives the same result up to presented precision.

Data soucre	λ	ℓ/d	a/d	b/d	$\theta/^\circ$
Coutanceau and Bouard [21]*	0.024	2.04	0.73	0.58	–
	0	2.13	0.76	0.59	53.5
Linnick and Fasel [22]	0.023	2.28	0.72	0.60	53.6
Calhoun [23]	–	2.18	–	–	54.2
Le et al. [24]	–	2.22	–	–	53.6
Taira and Colonius [25]	1/60	2.30	0.73	0.60	53.7
Berthelsen and Faltinsen [26]	–	2.29	0.72	0.60	53.9
present (direct/linear method)	1/30	2.27	0.72	0.59	53.1

Table 6
Coordinates of points of interest for the elliptic cylinder ($Re = 20$). Both direct and linear method gives the same result up to presented precision.

$\boldsymbol{a}^-/d$	$\boldsymbol{a}^+/d$	$b_\theta^-/^\circ$	$b_\theta^+/^\circ$	$\boldsymbol{c}^-/d$	$\boldsymbol{c}^+/d$	$\boldsymbol{d}/d$
$\begin{pmatrix} -0.11 \\ 0.08 \end{pmatrix}$	$\begin{pmatrix} 0.11 \\ -0.06 \end{pmatrix}$	0.3	178.7	$\begin{pmatrix} 0.78 \\ -0.33 \end{pmatrix}$	$\begin{pmatrix} 0.65 \\ 0.33 \end{pmatrix}$	$\begin{pmatrix} 1.97 \\ -0.02 \end{pmatrix}$

Table 7
Drag coefficients around the circular cylinder ($Re = 40$). Lift coefficients have been checked below 10^{-8} for the present study. Both direct and linear method gives the same result up to presented precision.

Data source	C_{Dp}	$C_{D\nu}$	C_D
Tritton [27]*			1.57
Linnick and Fasel [22]			1.54
Calhoun [23]			1.62
Le et al. [24]			1.56
Taira and Colonius [25]			1.54
Berthelsen and Faltinsen [26]			1.59
present (direct/linear method)	1.026	0.533	1.559

Table 8
Drag and lift coefficients around the elliptic cylinder inclined at 80° ($Re = 20$). Both direct and linear method gives the same result up to presented precision.

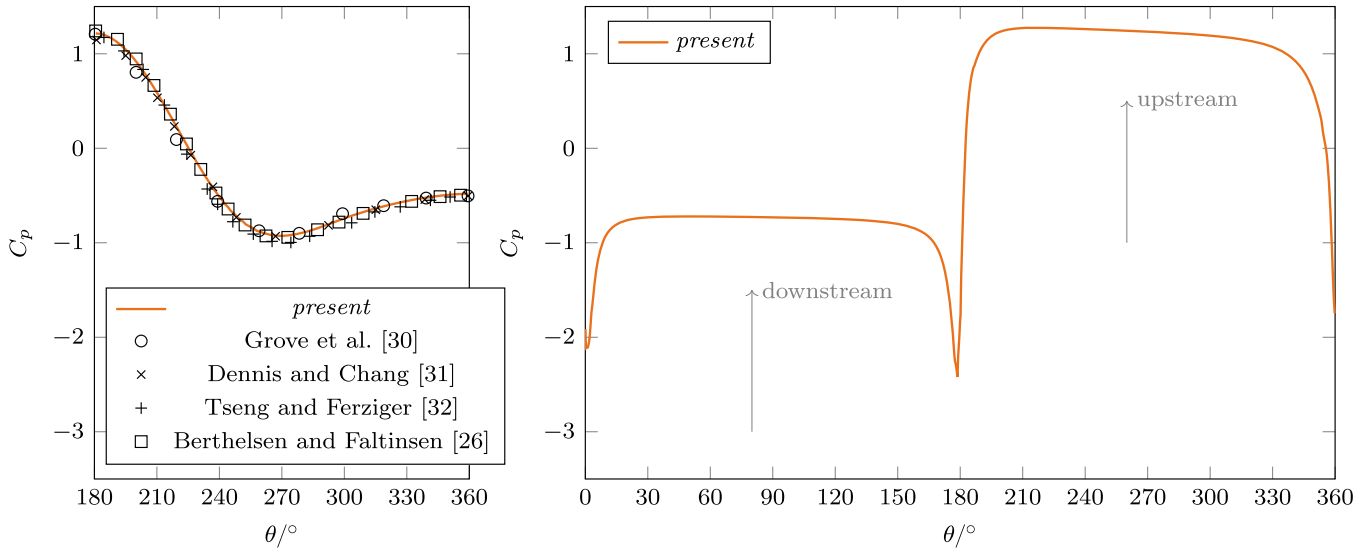
Data source	C_{Dp}	$C_{D\nu}$	C_D	C_{Lp}	$C_{L\nu}$	C_L
D'alessio and Dennis [28]			2.116			0.256
Dennis and Young [29]			2.089			0.255
Yoon et al. [20]			2.102			0.252
present (direct/linear method)	1.864	0.266	2.130	0.303	−0.057	0.246

8. Conclusions

We studied the size of the discretization stencil of the two Finite-Difference Ghost-Fluid Immersed Boundary methods defined in [3] and [4], that we called the linear and the direct method, respectively. We showed that the stencil size increases as cells become more and more rectangular, i.e. the aspect ratio of the cell increases, and we also showed that the direct method may become ill posed on rectangular cells. We proposed the Ghost Node Shifting Method to reduce the stencil size of any rectangular cell to the case of square cells, which also ensure the direct method to be well posed. We also showed that only the direct method is able to keep a stencil size of 1 while providing a second-order limiting behavior method with Dirichlet boundary conditions, and first-order limiting behavior method with Neumann boundary conditions.

We carried out numerical simulations demonstrating that shifted methods where able to achieve the desired limiting behavior, given an appropriate stencil size, see Table 2 for details. Simulations also showed that error levels of shifted methods are at least as low as error levels of original methods, and that the shifted direct method has less or the same error levels that the shifted linear method.

We also applied the shifted methods to the pressure-correction method to solve the incompressible Navier–Stokes problem with immersed boundaries. An emphasis has been given at each step of the pressure-correction method to produce a coherent solution with respect to the immersed boundaries: we showed that an additional extrapolation of the velocity field



(a) Circular cylinder ($Re = 40$). (b) Elliptic cylinder ($Re = 20$).

Fig. 26. Normalized pressure profile C_p around the cylinder in the $\Re'$ coordinate frame.

is necessary due to the nonlinear term. We were able to produce a method with a compact stencil, i.e. a stencil size is 1. Numerical simulation showed a second-order limiting behavior on the velocity and a first-to-second-order limiting behavior on the pressure. Finally we were able to perform numerical simulations of the flow around circular and elliptic cylinders with coherent results with respect to the literature.

Future works include applications of this method to other problems, studies of higher order discretization, in which the hollow case problem may become more cumbersome as interpolation stencil increases. An other work is therefore to find a robust handling of the hollow case problem. Finally, a customization of the linear system solver, such as performed by Coco and Russo [4] on multi-grid solvers, would improve the performances of the method on very large problems.

Acknowledgements

This study was carried out with financial support from the French National Research Agency (ANR) in the framework of the “Investments for the future” Programme IdEx Bordeaux (ANR-10-IDEX-03-02), Cluster of excellence CPU.

The authors wish to thank the Aquitaine Regional Council for the financial support towards a 432-processor cluster investment, located in the I2M laboratory. Computer time for this study was also provided by the computing facilities MCIA (Mesocentre de Calcul Intensif Aquitain) of the Université de Bordeaux and of the Université de Pau et des Pays de l’Adour.

Appendix A. Discrete operators for the incompressible Navier–Stokes problem

Let’s give a closer look to the discretization of Eq. (49)–(51) on Ω_i. For the sake of readability, we use an intuitive notation of indexes, e.g. $\mathsf{C} = (i, j)$ (center), $\mathsf{L} = (i-1, j)$ (left), etc. Depending which nodes are considered the exact definition changes; it is given in Fig. A.27.

In the prediction step Eq. (49a), the diffusion term $-\nu\Delta\boldsymbol{u}^*$ discretizes as in Sect. 2 on the face nodes, and becomes the matrix product $-\nu\boldsymbol{L}\boldsymbol{U}^*$

$$(\boldsymbol{L}U^*)_\mathsf{C} = \frac{U^*_\mathsf{L} - U^*_\mathsf{C}}{h_x^2} + \frac{U^*_\mathsf{R} - U^*_\mathsf{C}}{h_x^2} + \frac{U^*_\mathsf{B} - U^*_\mathsf{C}}{h_y^2} + \frac{U^*_\mathsf{T} - U^*_\mathsf{C}}{h_y^2} + \mathcal{O}(h^2)\,, \qquad \forall \mathsf{C} \in C_i^x\,, \tag{A.1a}$$

$$(\boldsymbol{L}V^*)_\mathsf{C} = \frac{V^*_\mathsf{B} - V^*_\mathsf{C}}{h_y^2} + \frac{V^*_\mathsf{T} - V^*_\mathsf{C}}{h_y^2} + \frac{V^*_\mathsf{L} - V^*_\mathsf{C}}{h_x^2} + \frac{V^*_\mathsf{R} - V^*_\mathsf{C}}{h_x^2} + \mathcal{O}(h^2)\,, \qquad \forall \mathsf{C} \in C_i^y\,. \tag{A.1b}$$

Just as in Sect. 2, some $U^*_\mathsf{L}, U^*_\mathsf{R}, U^*_\mathsf{B}, U^*_\mathsf{T}$, and some $V^*_\mathsf{L}, V^*_\mathsf{R}, V^*_\mathsf{B}, V^*_\mathsf{T}$ correspond to ghost nodes, and a boundary condition treatment must be addressed. The advection term $\nabla\cdot(\boldsymbol{u}^* \otimes \boldsymbol{u}^n)$ discretizes into the matrix product $\boldsymbol{A}(\boldsymbol{U}^n)\boldsymbol{U}^*$ combining second-order, centered interpolations and finite-differences

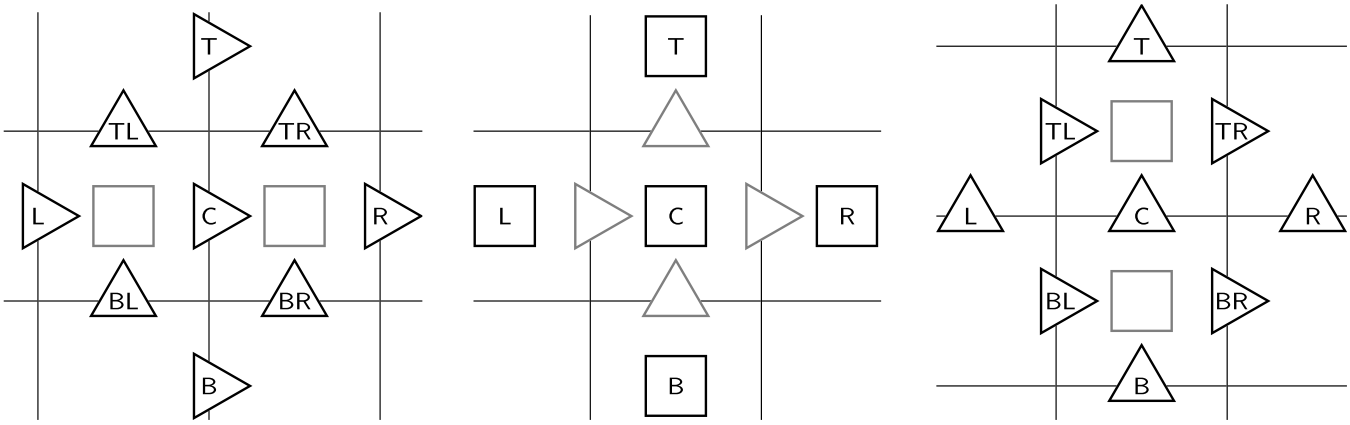


Fig. A.27. Relative node position used in Eq. (A.2a), (A.1a) (left), in Eq. (A.5) (center), and in Eq. (A.2b), (A.1b) (right).

$$\begin{aligned}\left(\boldsymbol{A}(\boldsymbol{U}^n)\boldsymbol{U}^*\right)_{\mathsf{C}} &= -\frac{U^n_{\mathsf{L}}+U^n_{\mathsf{C}}}{4h_x}U^*_{\mathsf{L}} + \frac{U^n_{\mathsf{R}}+U^n_{\mathsf{C}}}{4h_x}U^*_{\mathsf{R}} \\ &+ \left(-\frac{U^n_{\mathsf{L}}+U^n_{\mathsf{C}}}{4h_x} + \frac{U^n_{\mathsf{R}}+U^n_{\mathsf{C}}}{4h_x} - \frac{V^n_{\mathsf{BL}}+V^n_{\mathsf{BR}}}{4h_y} + \frac{V^n_{\mathsf{TL}}+V^n_{\mathsf{TR}}}{4h_y}\right)U^*_{\mathsf{C}} \qquad \forall \mathsf{C} \in C^x_i\,, \\ &+ -\frac{V^n_{\mathsf{BL}}+V^n_{\mathsf{BR}}}{4h_y}U^*_{\mathsf{B}} + \frac{V^n_{\mathsf{TL}}+V^n_{\mathsf{TR}}}{4h_y}U^*_{\mathsf{T}} + \mathcal{O}(h^2)\,,\end{aligned} \tag{A.2a}$$

$$\begin{aligned}\left(\boldsymbol{A}(\boldsymbol{U}^n)\boldsymbol{U}^*\right)_{\mathsf{C}} &= -\frac{U^n_{\mathsf{BL}}+U^n_{\mathsf{TL}}}{4h_x}V^*_{\mathsf{L}} + \frac{U^n_{\mathsf{BR}}+U^n_{\mathsf{TR}}}{4h_x}V^*_{\mathsf{R}} \\ &+ \left(-\frac{U^n_{\mathsf{BL}}+U^n_{\mathsf{TL}}}{4h_x} + \frac{U^n_{\mathsf{BR}}+U^n_{\mathsf{TR}}}{4h_x} - \frac{V^n_{\mathsf{B}}+V^n_{\mathsf{C}}}{4h_y} + \frac{V^n_{\mathsf{T}}+V^n_{\mathsf{C}}}{4h_y}\right)V^*_{\mathsf{C}} \qquad \forall \mathsf{C} \in C^y_i\,. \\ &+ -\frac{V^n_{\mathsf{B}}+V^n_{\mathsf{C}}}{4h_y}V^*_{\mathsf{B}} + \frac{V^n_{\mathsf{T}}+V^n_{\mathsf{C}}}{4h_y}V^*_{\mathsf{T}} + \mathcal{O}(h^2)\,,\end{aligned} \tag{A.2b}$$

Here again some $U^*_{\mathsf{L}}, U^*_{\mathsf{R}}, U^*_{\mathsf{B}}, U^*_{\mathsf{T}}$, and some $V^*_{\mathsf{L}}, V^*_{\mathsf{R}}, V^*_{\mathsf{B}}, V^*_{\mathsf{T}}$ correspond to ghost nodes, but also some $U^n_{\mathsf{L}}, U^n_{\mathsf{R}}, U^n_{\mathsf{B}}, U^n_{\mathsf{T}}, U^n_{\mathsf{BL}}$, $U^n_{\mathsf{TL}}, U^n_{\mathsf{BR}}, U^n_{\mathsf{TR}}$, and some $V^n_{\mathsf{L}}, V^n_{\mathsf{R}}, V^n_{\mathsf{B}}, V^n_{\mathsf{T}}, V^n_{\mathsf{BL}}, V^n_{\mathsf{TL}}, V^n_{\mathsf{BR}}, V^n_{\mathsf{TR}}$ does not correspond to inner nodes. Taking the assumption of Sect. 2.3 that the immersed boundary is not too hollow, none of theses values correspond to outer nodes and it is sufficient that $\boldsymbol{U}^n$ is being defined over C^{xy}_{ig}. The pressure gradient term $-\nabla p^n$ discretizes into the matrix product $-\boldsymbol{G}P^n$ using second-order centered finite-differences

$$(\boldsymbol{G}P^n)_{i+\frac{1}{2},j} = \frac{P^n_{i+1,j} - P^n_{i,j}}{h_x} + \mathcal{O}(h^2)\,, \qquad \forall (i,j) \in C^x_i\,, \tag{A.3a}$$

$$(\boldsymbol{G}P^n)_{i,j+\frac{1}{2}} = \frac{P^n_{i,j+1} - P^n_{i,j}}{h_y} + \mathcal{O}(h^2)\,, \qquad \forall (i,j) \in C^y_i\,, \tag{A.3b}$$

which requires that P^n is defined over C_i and some parts of C_g to be defined. More precisely, P^n must be defined on ghost cell nodes surrounded by a inner face node. In the correction step Eq. (51a), the right-hand side $\nabla\cdot\boldsymbol{u}^*$ discretizes into the matrix product $D\boldsymbol{U}^*$ using second-order, centered finite-differences

$$(D\boldsymbol{U}^*)_{i,j} = \frac{U^*_{i+\frac{1}{2},j} - U^*_{i-\frac{1}{2},j}}{h_x} + \frac{V^*_{i,j+\frac{1}{2}} - V^*_{i,j-\frac{1}{2}}}{h_y} + \mathcal{O}(h^2)\,, \qquad \forall (i,j) \in C_i\,. \tag{A.4}$$

The right-hand side is always defined over C_i as $\boldsymbol{U}^*$ is solved over C^{xy}_{ig} Eq. (52a). Finally the left-hand side $h_t\Delta\phi$ discretizes just as in Sect. 2 into the matrix product $h_t L\Phi + \mathcal{O}(h^2)$

$$(L\Phi)_{\mathsf{C}} = \frac{\Phi_{\mathsf{L}} - \Phi_{\mathsf{C}}}{h_x^2} + \frac{\Phi_{\mathsf{R}} - \Phi_{\mathsf{C}}}{h_x^2} + \frac{\Phi_{\mathsf{B}} - \Phi_{\mathsf{C}}}{h_y^2} + \frac{\Phi_{\mathsf{T}} - \Phi_{\mathsf{C}}}{h_y^2} + \mathcal{O}(h^2)\,, \qquad \forall (i,j) \in C_i\,. \tag{A.5}$$

Here again some $\Phi_{\mathsf{L}}, \Phi_{\mathsf{R}}, \Phi_{\mathsf{B}}, \Phi_{\mathsf{T}}$ correspond to ghost nodes, and a boundary condition treatment must be addressed.

Appendix B. Detailed parameters of used linear system solvers

The following parameters have been used for the *hypre* preconditioners anytime they have been used:

SMG preconditioner (all cases)

RelChange	1
ZeroGuess	*set*
NumPreRelax	1
NumPostRelax	1

BoomerAMG preconditioner (flower-shaped interface with Dirichlet boundary conditions)

StrongThreshold	0.25
CoarsenType	6, *Falgout coarsening*
AggNumLevels	1
RelaxType	6, *hybrid symmetric Gauss–Seidel or SSOR*
InterpType	0, *classical modified interpolation*

BoomerAMG preconditioner (circular interface with Neumann boundary conditions)

StrongThreshold	0.025
CoarsenType	10, *HMIS coarsening*
AggNumLevels	1
RelaxType	6, *hybrid symmetric Gauss–Seidel or SSOR*
InterpType	0, *classical modified interpolation*

References

[1] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. (ISSN 0021-9991) 10 (2) (1972) 252–271, https://doi.org/10.1016/0021-9991(72)90065-4.

[2] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (1) (2005) 239–261, https://doi.org/10.1146/annurev.fluid.37.061903.175743.

[3] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, J. Comput. Phys. (ISSN 0021-9991) 227 (10) (2008) 4825–4852, https://doi.org/10.1016/j.jcp.2008.01.028.

[4] A. Coco, G. Russo, Finite-difference ghost-point multigrid methods on Cartesian grids for elliptic problems in arbitrary domains, J. Comput. Phys. (ISSN 0021-9991) 241 (2013) 464–501, https://doi.org/10.1016/j.jcp.2012.11.047.

[5] S. Schaffer, A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients, SIAM J. Sci. Comput. 20 (1) (2006) 228–242, https://doi.org/10.1137/S1064827595281587.

[6] P.N. Brown, R.D. Falgout, J.E. Jones, Semicoarsening multigrid on distributed memory machines, SIAM J. Sci. Comput. (ISSN 1064-8275) 21 (5) (2000) 1823–1834, https://doi.org/10.1137/S1064827598339141.

[7] S.F. Ashbf, R.D. Falgout, A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations, Nucl. Sci. Eng. 126 (1996) 145–159.

[8] R.D. Falgout, J.E. Jones, Multigrid on massively parallel architectures, in: Multigrid Methods VI, Ghent, Belgium, 1999, pp. 101–107.

[9] U. Langer, D. Pusch, Comparison of geometrical and algebraic multigrid preconditioners for data-sparse boundary element matrices, in: International Conference on Large-Scale Scientific Computing, Springer, 2005, pp. 130–137.

[10] A.H. Baker, R.D. Falgout, T.V. Kolev, U.M. Yang, Scaling Hypre's multigrid solvers to 100,000 cores, in: M.W. Berry, K.A. Gallivan, E. Gallopoulos, A. Grama, B. Philippe, Y. Saad, F. Saied (Eds.), High-Performance Scientific Computing, Springer, London, ISBN 978-1-4471-2436-8, 2012, pp. 261–279, ISBN 978-1-4471-2437-5.

[11] E.H. Müller, R. Scheichl, Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction: scalability of elliptic solvers in NWP, Q. J. R. Meteorol. Soc. (ISSN 0035-9009) 140 (685) (2014) 2608–2624, https://doi.org/10.1002/qj.2327.

[12] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl. (ISSN 0895-4798) 23 (1) (2001) 15–41, https://doi.org/10.1137/S0895479899358194.

[13] P.R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, Hybrid scheduling for the parallel solution of linear systems, Parallel Comput. (ISSN 0167-8191) 32 (2) (2006) 136–156, https://doi.org/10.1016/j.parco.2005.07.004.

[14] R.D. Falgout, J.E. Jones, U.M. Yang, The design and implementation of hypre, a library of parallel high performance preconditioners, in: Numerical Solution of Partial Differential Equations on Parallel Computers, Springer, Berlin, Heidelberg, 2006, pp. 267–294.

[15] J.A. Mousel, A Massively Parallel Adaptive Sharp Interface Solver with Application to Mechanical Heart Valve Simulations, Ph.D. thesis, University of Iowa, 2012.

[16] Z. Li, A fast iterative algorithm for elliptic interface problems, SIAM J. Numer. Anal. 35 (1998) 230–254, https://doi.org/10.1137/S0036142995291329.

[17] F. Gibou, R.P. Fedkiw, L.-T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, J. Comput. Phys. (ISSN 0021-9991) 176 (1) (2002) 205–227, https://doi.org/10.1006/jcph.2001.6977.

[18] J.L. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Comput. Methods Appl. Mech. Eng. (ISSN 0045-7825) 195 (44–47) (2006) 6011–6045, https://doi.org/10.1016/j.cma.2005.10.010.

[19] L.J.P. Timmermans, P.D. Minev, F.N. Van De Vosse, An approximate projection scheme for incompressible flow using spectral elements, Int. J. Numer. Methods Fluids 22 (1996) 673–688.

[20] H.S. Yoon, J. Yin, C. Choi, S. Balachandar, M.Y. Ha, Bifurcation of laminar flow around an elliptic cylinder at incidence for low Reynolds numbers, Int. J. Prog. Comput. Fluid Dyn. 16 (3) (2016) 163–178.

[21] M. Coutanceau, R. Bouard, Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 2. Unsteady flow, J. Fluid Mech. (ISSN 1469-7645) 79 (2) (1977) 257–272, https://doi.org/10.1017/S0022112077000147, ISSN 0022-1120.

[22] M.N. Linnick, H.F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, J. Comput. Phys. (ISSN 0021-9991) 204 (1) (2005) 157–192, https://doi.org/10.1016/j.jcp.2004.09.017.

[23] D. Calhoun, A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions, J. Comput. Phys. (ISSN 0021-9991) 176 (2) (2002) 231–275, https://doi.org/10.1006/jcph.2001.6970.
[24] D. Le, B. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, J. Comput. Phys. (ISSN 0021-9991) 220 (1) (2006) 109–138, https://doi.org/10.1016/j.jcp.2006.05.004.
[25] K. Taira, T. Colonius, The immersed boundary method: a projection approach, J. Comput. Phys. (ISSN 0021-9991) 225 (2) (2007) 2118–2137, https://doi.org/10.1016/j.jcp.2007.03.005.
[26] P.A. Berthelsen, O.M. Faltinsen, A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries, J. Comput. Phys. (ISSN 0021-9991) 227 (9) (2008) 4354–4397, https://doi.org/10.1016/j.jcp.2007.12.022.
[27] D.J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds numbers, J. Fluid Mech. (ISSN 0022-1120) 6 (04) (1959) 547, https://doi.org/10.1017/S0022112059000829, ISSN 1469-7645.
[28] S.J.D. D'alessio, S.C.R. Dennis, A vorticity model for viscous flow past a cylinder, Comput. Fluids 23 (2) (1994) 279–293.
[29] S.C.R. Dennis, P.J.S. Young, Steady flow past an elliptic cylinder inclined to the stream, J. Eng. Math. 47 (2) (2003) 101–120.
[30] A.S. Grove, F.H. Shair, E.E. Petersen, An experimental investigation of the steady separated flow past a circular cylinder, J. Fluid Mech. (ISSN 0022-1120) 19 (1) (1964) 60, https://doi.org/10.1017/S0022112064000544, ISSN 1469-7645.
[31] S.C.R. Dennis, G.-Z. Chang, Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100, J. Fluid Mech. (ISSN 0022-1120) 42 (03) (1970) 471, https://doi.org/10.1017/S0022112070001428, ISSN 1469-7645.
[32] Y.-H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys. (ISSN 0021-9991) 192 (2) (2003) 593–623, https://doi.org/10.1016/j.jcp.2003.07.024.

5 Improvements on open and traction boundary conditions for Navier–Stokes time-splitting methods

Journal of Computational Physics 230 (2011) 4011–4027

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Improvements on open and traction boundary conditions for Navier–Stokes time-splitting methods

A. Poux, S. Glockner *, M. Azaïez

Université de Bordeaux, UMR CNRS 5295, IPB, 16 avenue Pey-Berland, 33607 Pessac, France

ARTICLE INFO

Article history:
Received 5 November 2010
Received in revised form 4 February 2011
Accepted 16 February 2011
Available online 22 February 2011

Keywords:
Navier–Stokes equations
Projection methods
Pressure-correction methods
Fractional step methods
Traction boundary condition
Open boundary condition
Pressure boundary condition
Bifurcated tube
Square cylinder

ABSTRACT

We present in this paper a numerical scheme for incompressible Navier–Stokes equations with open and traction boundary conditions, in the framework of pressure-correction methods. A new way to enforce this type of boundary condition is proposed and provides higher pressure and velocity convergence rates in space and time than found in the present state of the art. We illustrate this result by computing some numerical and physical tests. In particular, we establish reference solutions of a laminar flow in a geometry where a bifurcation takes place and of the unsteady flow around a square cylinder.

1. Introduction

The main difficulty in obtaining the numerical solution of the incompressible Navier–Stokes equations, apart from the treatment of non-linearities, lies in the Stokes stage and specifically in the determination of the pressure field which will ensure a solenoidal velocity field. The question is how to uncouple the velocity and the pressure operators to efficiently reach an accurate solution to the unsteady Stokes problem, without degrading the predefined stability properties of the chosen scheme for the Navier–Stokes equations.

Historically, the first idea was proposed by Uzawa [1,2] and applied for numerical approximations with several methods [3,4]. It is a safe and efficient method for the numerical approximation of the Stokes problem. In complex geometries or three-dimensional domains, this method turns out to be inappropriate for its computational time cost which becomes very high. A different method is to uncouple the pressure from the velocity by means of a time splitting scheme that significantly reduces the computational cost. A large number of theoretical and numerical works have been published that discuss the accuracy and the stability properties of such methods. The most widespread methods are pressure-correction schemes. They were first introduced by Chorin-Temam [5,6], and improved by Goda (the standard incremental scheme) in [7], and later by Timmermans in [8] (the rotational incremental scheme). They require the solution of two sub-steps for each time step: the pressure is treated explicitly in the first one, and is corrected in the second one by projecting the predicted velocity onto an ad hoc space. The governing equation on the pressure or the pressure increment is a Poisson equation derived from the

* Corresponding author.
E-mail addresses: apoux@enscbp.fr (A. Poux), glockner@enscbp.fr (S. Glockner), azaiez@enscbp.fr (M. Azaïez).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.02.024

momentum equation by requiring incompressibility. In [9,10], the authors proved the reliability of such approaches from the stability and the convergence rate point of view. A series of numerical issues related to the analysis and implementation of fractional step methods for incompressible flows are addressed in the review paper of Guermond et al. [11]. In this reference, the authors describe the state of the art for both theoretical and numerical results related to the time splitting approach. One emerging conclusion points out that time splitting can be a high order alternative to solve the unsteady Stokes problem when the velocity boundary condition is of the Dirichlet type.

However, in many applications such as free surface problems and channel flows, one also has to deal with an outlet boundary condition on all or part of the boundary, on which the applied numerical condition should not disturb upstream flow. A large variety of this kind of boundary condition exists [12,13], such as the non-reflecting outlet boundary condition (and its adaptations) derived from a wave equation, which is suited to wake and jet flow with moderate and high Reynolds number [14–19]. Hereafter we will present some improvements on the open or traction boundary condition which is efficient for low Reynolds number and fluid–structure interactions [20–22]. The traction boundary condition was successfully used to compute various flows such as those around a circular cylinder, over a backward facing step and in a bifurcated tube [20]. Bruneau [23] proposes an evolution of the traction boundary condition involving inertial terms. Hasan [24] proposes, in the computation of incompressible flow around rigid bodies, to extrapolate velocity on the outflow boundary, pressure being obtained through traction boundary conditions.

In the case of open or traction boundary conditions, several questions remain open especially when the pressure-correction version is considered as mentioned in [10]. Indeed, Guermond et al. [11], have proven that only spatial and time convergence rates between $O(\Delta x + \Delta t)$ and $O(\Delta x^{3/2} + \Delta t^{3/2})$ on the velocity and $O(\Delta x^{1/2} + \Delta t^{1/2})$ on the pressure are to be expected with the standard incremental scheme, and between $O(\Delta x + \Delta t)$ and $O(\Delta x^{3/2} + \Delta t^{3/2})$ on the velocity and pressure for the rotational incremental scheme. Février [25] combines the penalty and projection methods to improve error levels of a manufactured case with open boundary conditions, but without improvement of the convergence rate. Finally, Liu [20], with a pressure Poisson equation formulation, proposes a new implementation of the open and traction boundary conditions. He proves unconditional stability with a first order time scheme and shows second order numerical convergence rate on velocity and pressure.

The aim of this paper is to propose a numerical scheme for the incompressible Navier–Stokes equations with open and traction boundary conditions, using the pressure-correction version of the time splitting methods. A new way to enforce this type of boundary condition is proposed and improves the order of convergence for both pressure and velocity. In the second part of this article we will describe the governing equations, and, in the third part, the pressure-correction schemes for open boundary conditions. In the fourth part, we will present the improvements we made on the numerical implementation of the traction and open boundary conditions. In a fifth section we will illustrate numerically the proposed method with two manufactured cases and two physical cases.

First of all we specify some notations. Let us consider a Lipschitz domain $\Omega \subset \mathbb{R}^d$, $(d = 2 \text{ or } 3)$, the generic point of Ω is denoted $\mathbf{x}$. The classical Lebesgue space of square integrable functions $L^2(\Omega)$ is endowed with the inner product:

$$(\phi, \psi) = \int_\Omega \phi\psi \; d\boldsymbol{x},$$

and the norm

$$\|\psi\|_{L^2(\Omega)} = \left(\int_\Omega |\psi(\boldsymbol{x})|^2 \right)^{\frac{1}{2}}.$$

We break the time interval $[0, t^*]$ into N subdivisions of length $\Delta t = \frac{t^*}{N}$, called the time step, and define $t^n = n\Delta t$, for any n, $0 \leqslant n \leqslant N$. Let $\varphi^0, \varphi^1, \ldots, \varphi^N$ be some sequence of functions in $E = L^2$. We denote this sequence by $\varphi^{\Delta t}$, and we define the following discrete norm

$$\|\varphi^{\Delta t}\|_{l^2(E)} = \left(\Delta t \sum_{k=0}^{N} \|\varphi^k\|_E^2 \right)^{\frac{1}{2}}. \tag{1.1}$$

In practice the following error estimator can be used

$$\|\varphi\|_{E(t^*)}^2 = \|\varphi(\cdot, t^*)\|_E. \tag{1.2}$$

Finally, bold Latin letters like $\boldsymbol{u}$, $\boldsymbol{w}$, $\boldsymbol{f}$, etc., indicate vector valued quantities, while capitals (e.g. $\boldsymbol{X}$, etc.) are functional sets involving vector fields.

2. Governing equations

Let Ω be a regular bounded domain in $\mathbb{R}^d$ with $\boldsymbol{n}$ the unit normal to the boundary $\Gamma = \partial\Omega$ oriented outward. We suppose that Γ is partitioned into two portions Γ_D and Γ_N.

Our study consists, for a given finite time interval $[0, t^*]$ in computing velocity $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t)$ and pressure $p = p(\boldsymbol{x}, t)$ fields satisfying:

$$\rho\partial_t \boldsymbol{u} - \mu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \quad \text{in } \Omega\times[0,t^*], \tag{2.3}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega\times[0,t^*], \tag{2.4}$$

$$\boldsymbol{u} = \boldsymbol{g} \quad \text{on } \Gamma_D\times[0,t^*], \tag{2.5}$$

$$(\mu\nabla\boldsymbol{u} - p\boldsymbol{Id})\boldsymbol{n} = \boldsymbol{t} \quad \text{on } \Gamma_N\times[0,t^*], \tag{2.6}$$

where ρ and μ are respectively the density and the dynamic viscosity of the flow. The body force $\boldsymbol{f}=\boldsymbol{f}(\boldsymbol{x},t)$, the pseudo-constraint $\boldsymbol{t}=\boldsymbol{t}(\boldsymbol{x},t)$ and the boundary condition $\boldsymbol{g}=\boldsymbol{g}(\boldsymbol{x},t)$ are known. For sake of simplicity, we chose $\boldsymbol{g}=0$. Finally, the initial state is characterised by a given $\boldsymbol{u}(.,0)$.

In some cases, the open or natural boundary condition (2.6) can be replaced by the traction boundary condition written as

$$\left(\mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) - p\boldsymbol{Id}\right)\boldsymbol{n} = \boldsymbol{t}. \tag{2.7}$$

3. Pressure-correction schemes for open boundary condition

We shall compute two sequences $(\boldsymbol{u}^n)_{0\leqslant n\leqslant N}$ and $(p^n)_{0\leqslant n\leqslant N}$ in a recurrent way that approximate in some sense the quantities $(\boldsymbol{u}(.,t^n))_{0\leqslant n\leqslant N}$ and $(p(.,t^n))_{0\leqslant n\leqslant N}$, solutions of the unsteady Stokes problem (2.3)–(2.6). Its semi-discrete version reads:

$$\rho\frac{\alpha\boldsymbol{u}^{n+1}+\beta\boldsymbol{u}^n+\gamma\boldsymbol{u}^{n-1}}{\Delta t} - \mu\Delta\boldsymbol{u}^{n+1} + \nabla p^{n+1} = \boldsymbol{f}^{\,n+1} \quad \text{in } \Omega, \tag{3.8}$$

$$\nabla\cdot\boldsymbol{u}^{n+1} = 0 \quad \text{in } \Omega, \tag{3.9}$$

$$\mathbf{u}^{n+1} = 0 \quad \text{on } \Gamma_D, \tag{3.10}$$

$$(\mu\nabla\boldsymbol{u}^{n+1} - p^{n+1}\boldsymbol{Id})\boldsymbol{n} = \boldsymbol{t}^{n+1} \quad \text{on } \Gamma_N. \tag{3.11}$$

Values of parameters α, β, γ depend on the temporal scheme used. Namely:

- $\alpha=1$, $\beta=-1$, $\gamma=0$ for the first order Euler time scheme,
- $\alpha=\frac{3}{2}$, $\beta=-2$, $\gamma=\frac{1}{2}$ for the second order Backward Difference Formulae time scheme.

Eqs. (3.8)–(3.11) are split into two subproblems. The first is the prediction diffusion problem that computes a prediction velocity fields: *find* $\boldsymbol{u}^{n+1/2}$ *such that:*

$$\rho\frac{\alpha\boldsymbol{u}^{n+1/2}+\beta\boldsymbol{u}^n+\gamma\boldsymbol{u}^{n-1}}{\Delta t} - \mu\Delta\boldsymbol{u}^{n+1/2} + \nabla p^n = \boldsymbol{f}^{\,n+1} \quad \text{in } \Omega, \tag{3.12}$$

$$\mathbf{u}^{n+1/2} = 0 \quad \text{on } \Gamma_D, \tag{3.13}$$

$$(\mu\nabla\boldsymbol{u}^{n+1/2} - \tilde{p}^{n+1}\boldsymbol{Id})\boldsymbol{n} = \boldsymbol{t}^{n+1} \quad \text{on } \Gamma_N. \tag{3.14}$$

In the last Eq. (3.14), the expression of $\tilde{p}^{n+1}$ depends on the considered time scheme. To ensure the expected order of the time approximation, we propose two cases:

- $\alpha=1$, $\beta=-1$, $\gamma=0$ then $\tilde{p}^{n+1}=p^n$,
- $\alpha=\frac{3}{2}$, $\beta=-2$, $\gamma=\frac{1}{2}$ then $\tilde{p}^{n+1}=2p^n-p^{n-1}$.

The second step is a pressure-continuity correction: *find* $(\boldsymbol{u}^{n+1},p^{n+1})$ *such that:*

$$\frac{\rho\alpha}{\Delta t}\left(\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n+1/2}\right) + \nabla\varphi^{n+1} = \boldsymbol{0} \quad \text{in } \Omega, \tag{3.15}$$

$$\nabla\cdot\boldsymbol{u}^{n+1} = 0 \quad \text{in } \Omega, \tag{3.16}$$

$$\boldsymbol{u}^{n+1}\cdot\boldsymbol{n} = 0 \quad \text{on } \Gamma_D, \tag{3.17}$$

$$\text{B.C.}^{\,n+1} \quad \text{on } \Gamma_N. \tag{3.18}$$

and the pressure is upgraded via:

$$p^{n+1} = p^n + \varphi^{n+1} - \chi\mu\nabla\cdot\boldsymbol{u}^{n+1/2} \quad \text{in } \Omega. \tag{3.19}$$

The parameter χ is used to switch between the standard incremental scheme and the rotational one:

- $\chi=0$ for the standard incremental scheme,

- $\chi = 0.7$ for the rotational incremental scheme[1].

The second step is rewritten as a Poisson problem on φ^{n+1}:

$$\frac{\Delta t}{\alpha\rho}\nabla\cdot\nabla\varphi^{n+1} = \nabla\cdot\boldsymbol{u}^{n+1/2} \quad \text{in } \Omega, \tag{3.20}$$

$$\frac{\partial}{\partial n}\varphi^{n+1} = 0 \quad \text{on } \Gamma_D, \tag{3.21}$$

$$\text{B.C.}^{n+1} \quad \text{on } \Gamma_N, \tag{3.22}$$

completed by:

$$p^{n+1} = p^n + \varphi^{n+1} - \chi\mu\nabla\cdot\boldsymbol{u}^{n+1/2} \quad \text{in } \Omega, \tag{3.23}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{n+1/2} - \frac{\Delta t}{\rho\alpha}\nabla\varphi^{n+1} \quad \text{in } \Omega. \tag{3.24}$$

The main result of this contribution lies in the definition of B.C.$^{n+1}$ in (3.22). As mentioned in [22] the "natural choice" which consists in considering φ^{n+1} equals zero on Γ_N leads to a kind of numerical locking for $\chi = 0$ since the boundary condition on the pressure increment causes the pressure on the limit to be equal to its initial value. The convergence of the algorithm is low and slow. To circumvent this difficulty the authors suggest to use the rotational incremental version. In the present work we propose an alternative improving the accuracy for the standard incremental version while remaining compatible with the rotational one. Indeed, the boundary condition on φ can be induced by derivation of the Helmholtz–Hodge decomposition of the $\boldsymbol{u}^{n+1/2}$ (3.15). This approach has already been followed by Brazza in [17] and Kirpatrick in [26] for other kinds of outlet boundary conditions (non-reflecting or Neumann).

4. Improvement of the pressure boundary conditions

For the sake of simplicity we choose Ω to be a square and we fix Γ_N at its right boundary. The starting point of our approach is the first component of Eq. (3.15) that we derive on x_1:

$$-\frac{\Delta t}{\alpha\rho}\partial_{x_1^2}\varphi^{n+1} = \partial_{x_1}u_{x_1}^{n+1} - \partial_{x_1}u_{x_1}^{n+\frac{1}{2}}. \tag{4.25}$$

We project Eqs. (3.11) and (3.14) on direction x_1:

$$\mu\partial_{x_1}u_{x_1}^{n+1} - p^{n+1} = t_{x_1}^{n+1}, \tag{4.26}$$

$$\mu\partial_{x_1}u_{x_1}^{n+\frac{1}{2}} - \tilde{p}^{n+1} = t_{x_1}^{n+1}, \tag{4.27}$$

and combining those three equations, one can verify that (4.25) is equivalent to:

$$-\frac{\Delta t}{\alpha\rho}\partial_{x_1^2}\varphi^{n+1} = \frac{1}{\mu}(p^{n+1} - \tilde{p}^{n+1}). \tag{4.28}$$

Depending on the choice of the time splitting version and the time order scheme, $p^{n+1} - \tilde{p}^{n+1}$ can be replaced by its approximation expression using φ^{n+1} and the divergence of the predicted velocity. For the first order time scheme, we easily derive the following boundary conditions:

$$\left(\frac{\Delta t}{\alpha\rho}\partial_{x_1^2} + \frac{1}{\mu}\right)\varphi^{n+1} = \chi\nabla\cdot\boldsymbol{u}^{n+\frac{1}{2}}. \tag{4.29}$$

Or, for a second order scheme:

$$\left(\frac{\Delta t}{\alpha\rho}\partial_{x_1^2} + \frac{1}{\mu}\right)\varphi^{n+1} = \frac{\varphi^n}{\mu} + \chi\nabla\cdot\left(\boldsymbol{u}^{n+\frac{1}{2}} - \boldsymbol{u}^{n-\frac{1}{2}}\right). \tag{4.30}$$

Subtracting (4.29) or (4.30) in (3.20) leads to a version that is easier to implement:

- First-order open boundary condition (OBC1):

$$\left(\frac{\Delta t}{\alpha\rho}\partial_{x_2^2} - \frac{1}{\mu}\right)\varphi^{n+1} = (1-\chi)\nabla\cdot\boldsymbol{u}^{n+\frac{1}{2}}. \tag{4.31}$$

[1] Ideally, $\chi = 1$ but as Guermond proved in [22], for stability issues, χ is necessarily strictly less than $2\mu/d$.

- Second-order open boundary condition (OBC2):

$$\left(\frac{\Delta t}{\alpha\rho}\partial_{x_2^2} - \frac{1}{\mu}\right)\varphi^{n+1} = (1-\chi)\nabla\cdot\boldsymbol{u}^{n+\frac{1}{2}} - \frac{\varphi^n}{\mu} + \chi\nabla\cdot\boldsymbol{u}^{n-\frac{1}{2}}. \tag{4.32}$$

In summary, our approach differs from the usual one by the definition of the boundary condition on Γ_N in the correction step.

We end this section by describing the extension of (4.31) and (4.32) to a more general partition of the boundary $\partial\Omega$.

4.1. Generalisation of the boundary condition

Extension to a more general definition of Γ_N requires new notations to be introduced. Let us denote by $\boldsymbol{D}(\boldsymbol{u}) = (\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)/2$, the rate of deformation tensor and $\boldsymbol{\sigma}(\boldsymbol{u},p) = -p\boldsymbol{Id} + 2\mu\boldsymbol{D}$ the stress one. $\boldsymbol{n}$ and $\boldsymbol{\tau}$ are respectively the units normal to the boundary oriented outward and the associated unit tangent vector.

The traction boundary condition $\boldsymbol{\sigma}(\boldsymbol{u},p)$: $\boldsymbol{n} = \boldsymbol{t}$ projected on $\boldsymbol{n}$ and $\boldsymbol{\tau}$ is written:

$$2\mu\boldsymbol{D}(\boldsymbol{u}^{n+1}) : \boldsymbol{n}\cdot\boldsymbol{n} - p^{n+1} = \boldsymbol{t}^{n+1}\cdot\boldsymbol{n}, \tag{4.33}$$

$$2\mu\boldsymbol{D}(\boldsymbol{u}^{n+1}) : \boldsymbol{n}\cdot\boldsymbol{\tau}. = \boldsymbol{t}^{n+1}\cdot\boldsymbol{\tau}. \tag{4.34}$$

Applying $\boldsymbol{D}$ to (3.15) gives us:

$$\boldsymbol{D}(\boldsymbol{u}^{n+1}) - \boldsymbol{D}\left(\boldsymbol{u}^{n+\frac{1}{2}}\right) = -\frac{\Delta t}{\alpha\rho}\boldsymbol{D}(\nabla\varphi^{n+1}). \tag{4.35}$$

Denoting $\boldsymbol{L}(\varphi^{n+1}) = \boldsymbol{D}(\nabla\varphi^{n+1})$, we have:

$$\boldsymbol{D}(\boldsymbol{u}^{n+1}) : \boldsymbol{n}\cdot\boldsymbol{n} - \boldsymbol{D}\left(\boldsymbol{u}^{n+\frac{1}{2}}\right) : \boldsymbol{n}\cdot\boldsymbol{n} = -\frac{\Delta t}{\alpha\rho}\boldsymbol{L}(\varphi^{n+1}) : \boldsymbol{n}\cdot\boldsymbol{n}. \tag{4.36}$$

Using (4.33) and (3.23) we obtain:

$$-\frac{\Delta t}{\alpha\rho}\boldsymbol{L}(\varphi^{n+1}) : \boldsymbol{n}\cdot\boldsymbol{n} = \frac{\boldsymbol{t}^{n+1}\cdot\boldsymbol{n} + p^n + \varphi^{n+1} - \chi\mu\nabla\cdot\boldsymbol{u}^{n+\frac{1}{2}}}{2\mu} - \boldsymbol{D}\left(\boldsymbol{u}^{n+\frac{1}{2}}\right) : \boldsymbol{n}\cdot\boldsymbol{n}.$$

Thus, for the first order boundary condition, we can impose on $(\boldsymbol{u}^{n+\frac{1}{2}}, \varphi^{n+1})$:

$$2\mu\boldsymbol{D}\left(\boldsymbol{u}^{n+\frac{1}{2}}\right) : \boldsymbol{n}\cdot\boldsymbol{n} = \boldsymbol{t}^{n+1}\cdot\boldsymbol{n} + p^n, \tag{4.37}$$

$$2\mu\boldsymbol{D}\left(\boldsymbol{u}^{n+\frac{1}{2}}\right) : \boldsymbol{n}\cdot\boldsymbol{\tau} = \boldsymbol{t}^{n+1}\cdot\boldsymbol{\tau}, \tag{4.38}$$

$$\frac{\Delta t}{\alpha\rho}\boldsymbol{L}(\varphi^{n+1}) : \boldsymbol{n}\cdot\boldsymbol{n} + \frac{\varphi^{n+1}}{2\mu} = \frac{\chi}{2}\nabla\cdot\boldsymbol{u}^{n+\frac{1}{2}}. \tag{4.39}$$

Or, to have second order scheme:

$$2\mu\boldsymbol{D}\left(\boldsymbol{u}^{n+\frac{1}{2}}\right) : \boldsymbol{n}\cdot\boldsymbol{n} = \boldsymbol{t}^{n+1}\cdot\boldsymbol{n} + 2p^n - p^{n-1}, \tag{4.40}$$

$$2\mu\boldsymbol{D}\left(\boldsymbol{u}^{n+\frac{1}{2}}\right) : \boldsymbol{n}\cdot\boldsymbol{t} = \boldsymbol{t}^{n+1}\cdot\boldsymbol{\tau}, \tag{4.41}$$

$$\frac{\Delta t}{\alpha\rho}\boldsymbol{L}(\varphi^{n+1}) : \boldsymbol{n}\cdot\boldsymbol{n} + \frac{\varphi^{n+1}}{2\mu} = \frac{\varphi^n}{2\mu} + \frac{\chi}{2}\nabla\cdot\left(\boldsymbol{u}^{n+\frac{1}{2}} - \boldsymbol{u}^{n-\frac{1}{2}}\right). \tag{4.42}$$

To conclude this section, the full time-splitting algorithm in the presence of traction boundary conditions reads:

Prediction step. *Find* $\boldsymbol{u}^{n+1/2}$ *such that:*

$$\rho\frac{\alpha\boldsymbol{u}^{n+1/2} + \beta\boldsymbol{u}^n + \gamma\boldsymbol{u}^{n-1}}{\Delta t} - \mu\Delta\boldsymbol{u}^{n+1/2} + \nabla p^n = \boldsymbol{f}^{\,n+1}, \quad \text{in } \Omega, \tag{4.43}$$

$$\mathbf{u}^{n+1/2} = 0, \quad \text{on } \Gamma_D, \tag{4.44}$$

$$\left(\mu\left(\nabla\boldsymbol{u}^{n+1/2} + (\nabla\boldsymbol{u}^{n+1/2})^T\right) - \tilde{p}^{n+1}\boldsymbol{Id}\right)\boldsymbol{n} = \boldsymbol{t}^{n+1} \quad \text{on } \Gamma_N. \tag{4.45}$$

Correction pressure-continuity step. *Find* $(\boldsymbol{u}^{n+1}, p^{n+1})$ *such that:*

$$\frac{\Delta t}{\alpha\rho}\nabla\cdot\nabla\varphi^{n+1} = \nabla\cdot\boldsymbol{u}^{n+1/2} \quad \text{in } \Omega, \tag{4.46}$$

$$\frac{\partial}{\partial n}\varphi^{n+1} = 0 \quad \text{on } \Gamma_D, \tag{4.47}$$

$$\frac{\Delta t}{\alpha\rho}\boldsymbol{L}(\varphi^{n+1}):\boldsymbol{n}\cdot\boldsymbol{n} + \frac{\varphi^{n+1}}{2\mu} = \frac{\chi}{2}\nabla\cdot\boldsymbol{u}^{n+\frac{1}{2}} + \gamma\left(\frac{\varphi^n}{\mu} - \chi\nabla\cdot\boldsymbol{u}^{n-\frac{1}{2}}\right), \quad \text{on } \Gamma_N. \tag{4.48}$$

Completed by:

$$p^{n+1} = p^n + \varphi^{n+1} - \chi\mu\nabla\cdot\boldsymbol{u}^{n+1/2} \quad \text{in } \Omega, \tag{4.49}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{n+1/2} - \frac{\Delta t}{\rho\alpha}\nabla\varphi^{n+1} \quad \text{in } \Omega. \tag{4.50}$$

The choice of χ permits us to switch from the standard incremental scheme ($\chi = 0$) to the rotational form $\left(\chi < \frac{2\mu}{d}\right)$ while the values of α, β, γ move from the first time order to the second:

- $\alpha = 1$, $\beta = -1$, $\gamma = 0$ for first order Euler time scheme; then $\tilde{p}^{n+1} = p^n$,
- $\alpha = \frac{3}{2}$, $\beta = -2$, $\gamma = \frac{1}{2}$ for second order Backward Difference Formulae time scheme; then $\tilde{p}^{n+1} = 2p^n - p^{n-1}$.

5. Numerical experiments

We start this section by giving a brief description of the space discretisation and tools for the numerical simulations. Then, numerical experiments involving the proposed boundary conditions are separated into verification cases on manufactured solutions of the Stokes equations, and validation on physical cases governed by the Navier–Stokes equations.

5.1. Spatial discretisation and linear solvers

Spatial discretisation is based on the finite volume method. To avoid the well known spurious modes phenomena, the correction step uses the stable finite volume staggered grid of the Marker and Cells type [27]. In our implementation, pressure unknowns are associated to the cell vertices whereas velocity components are face centred. The approximation of the prediction-diffusion of the Stokes problem step uses a centred scheme of second order to compute the predicted velocity. The approach adopted for the treatment of the Navier–Stokes non-linear term $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ involved in the material derivative of the velocity consists in linearising it by $((2\boldsymbol{u}^{\boldsymbol{n}} - \boldsymbol{u}^{\boldsymbol{n-1}})\cdot\nabla)\boldsymbol{u}^{n+1}$ and applying a second-order centred scheme.

In the following simulations, the proposed boundary condition (4.48), which contains high order derivatives in the normal direction of the boundary, is treated by subtracting it in (3.20) which leads to a formulation involving only tangential derivatives. As pressure unknowns are associated to the cell vertices, these derivatives are trivial to discretise by a second-order centred scheme. This approach would be valid for any orthogonal curvilinear coordinate system.

All the computations are made using the parallel version of a Navier–Stokes solver based on the block-structured mesh partitioner described in [28]. Finally and in order to solve the linear systems we are using:

- For the prediction step, the iterative BiCGStab method with a point Jacobi preconditioner of the HYPRE library [29].
- For the correction step, the multifrontal sparse direct solver MUMPS [30] for manufactured cases or the GMRES method with a semi-coarsening geometric multigrid preconditioner [31,32] of the HYPRE library for physical cases.

5.2. Numerical results for the Stokes problem

To assess the error convergence rate of the methods in space and time, we consider two manufactured test cases with $\rho = 1$ and $\mu = 1$. The domain is chosen to be $\Omega = [-1;1]^2$ and we enforce the traction boundary condition on the right part of the box and a Dirichlet boundary condition everywhere else. Exact solutions for $\boldsymbol{u}^{ex} = (u^{ex}_{x_1}, u^{ex}_{x_2})$ and p^{ex} are:

- For the homogeneous case ($\boldsymbol{t} = \boldsymbol{0}$):

$$u^{ex}_{x_1} = \sin(x_1)\sin(x_2)\cos(2\pi\omega t),$$
$$u^{ex}_{x_2} = \cos(x_1)\cos(x_2)\cos(2\pi\omega t),$$
$$p^{ex} = -2\cos(1)\sin(2(x_1-1)-x_2)\cos(2\pi\omega t).$$

- For the non-homogeneous case, we take the same problem as [20]:

$$u_{x_1}^{ex} = \cos^2\left(\frac{\pi x_1}{4}\right)\sin\left(\frac{\pi x_2}{2}\right)\cos(2\pi\omega t),$$
$$u_{x_2}^{ex} = -\cos^2\left(\frac{\pi x_2}{4}\right)\sin\left(\frac{\pi x_1}{2}\right)\cos(2\pi\omega t),$$
$$p^{ex} = \cos\left(\frac{\pi x_1}{4}\right)\sin\left(\frac{\pi x_2}{4}\right)\cos(2\pi\omega t).$$

We present hereafter a convergence study with a traction boundary condition, homogeneous or not. The error estimator $\|\varphi\|^2_{L^2(t^*)}$ defined in (1.2) is used.

5.2.1. Error convergence rate in space

Most of the research effort made on pressure-correction methods deals with the time convergence rate, since optimal space one can be reached in presence of Dirichlet boundary conditions on velocity. Nevertheless, in open or traction boundary conditions, spatial convergence rates are limited to Δx for velocity and $O(\Delta x^{1/2})$ for pressure. The only known way to overcome this difficulty in the framework of pressure-correction methods is to switch to the rotational formulation.

In order to study the spatial splitting error we carried out the first numerical experiment with $\omega = 0$. The time step is fixed at $\Delta t = 10^{-3}$ and we run the algorithm for different values of Δx. Stationary tolerance is set to 10^{-12}. As long as we attempt to approximate analytical solutions by a second order scheme, we expect that the errors decay like Δx^2. The representative curves of Figs. 1 and 2 (on the left) show that the convergence rate of the error on the velocity and the pressure are $O(\Delta x^2)$. We obtain exactly the same result with the first and the second order (in time) versions.

Repeating these tests with the rotational scheme, we obtain a $O(\Delta x^2)$ convergence rate of the error on velocity and pressure whatever the method used (see Figs. 1 and 2 on the right).

As a conclusion on the spatial convergence rate, we can say that the proposed method improves the standard incremental scheme from $O(\Delta x)$ to $O(\Delta x^2)$ for the velocity and from $O(\Delta x^{1/2})$ to $O(\Delta x^2)$ for pressure, while remaining compatible with the rotational scheme. The question now is to know if the numerical locking of the incremental scheme also observed for the error convergence rate in time is overcome, and if the method is still compatible with the rotational formulation.

5.2.2. Error convergence rate in time

To study the time splitting error, we consider the unsteady case $\omega = 1$. In Figs. 3 and 4 we plotted for Δx fixed at 1/256 and for a range of time steps $10^{-4} \leqslant \Delta t \leqslant 10^{-1}$ the errors (1.2) at $t = 2$. With the first order proposed boundary condition, the convergence rate of the error on the velocity and the pressure are around $O(\Delta t)$. While with the second order boundary condition we ensure a convergence rate of $O(\Delta t^2)$, for the velocity and between $O(\Delta t^{3/2})$ and $O(\Delta t^2)$ for pressure. Thus, the convergence rate in the presence of an open or traction boundary condition is now brought to the level observed with the Dirichlet boundary condition.

We repeated these tests with the rotational scheme. As one can see in Figs. 5 and 6, for the standard boundary condition ($\varphi = 0$ on Γ_N) the convergence rate of the error on the velocity and the pressure is between $O(\Delta\, t^{3/2})$ and $O(\Delta t^2)$, which are the same as [22]. With the proposed first order boundary condition, the convergence rate of the error on the velocity and the pressure is around $O(\Delta t)$. And finally, with the second order boundary condition we can obtain a clear convergence rate of the error on the velocity of $O(\Delta t^2)$ and on the pressure of $O(\Delta t^2)$.

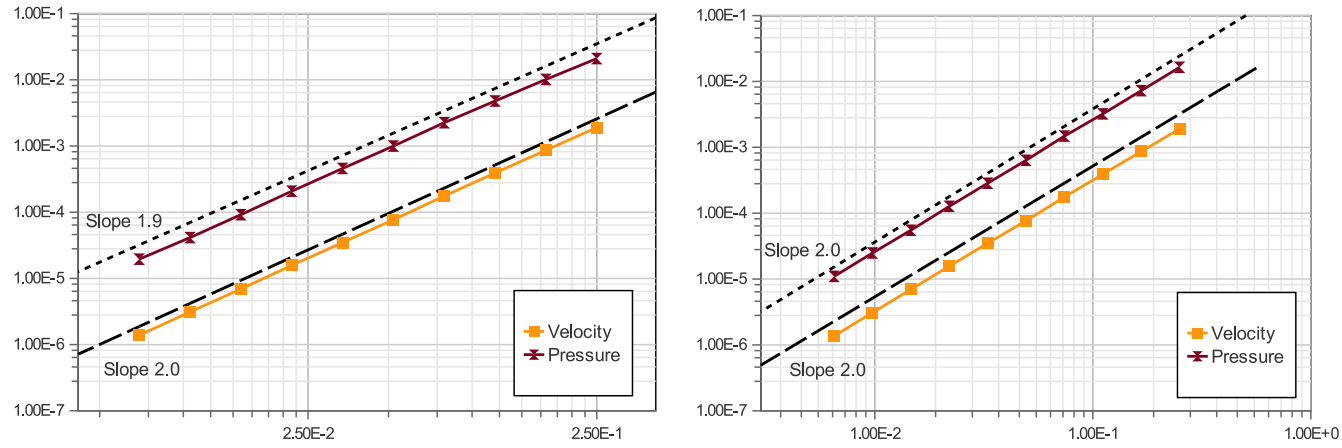


Fig. 1. Spatial convergence rates with the standard incremental scheme (left) and the rotational scheme (right) with $\Delta t = 10^{-3}$ for the steady homogeneous case.

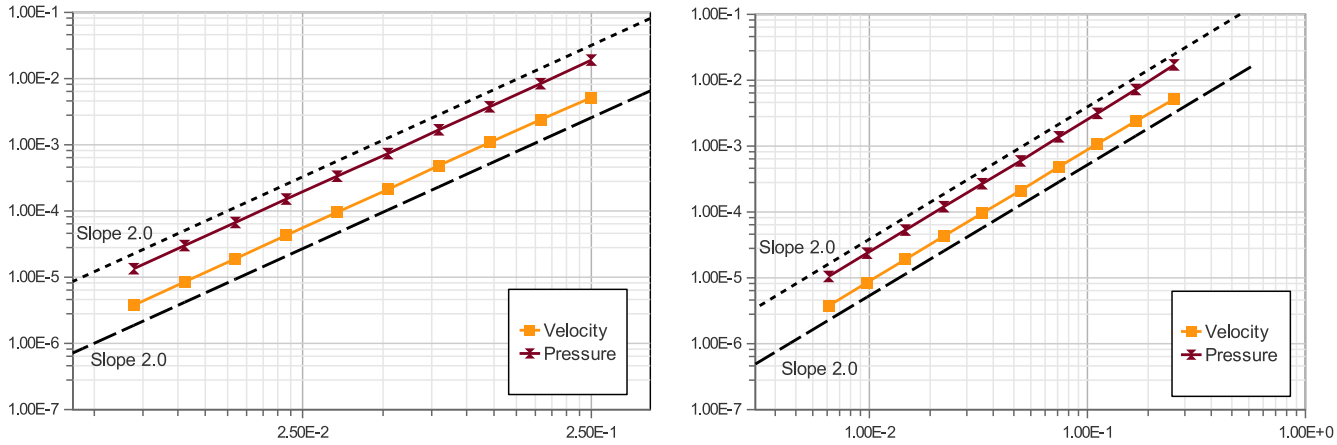


Fig. 2. Spatial convergence rates with the standard incremental scheme (left) and the rotational scheme (right) with $\Delta t = 10^{-3}$ for the steady non-homogeneous case.

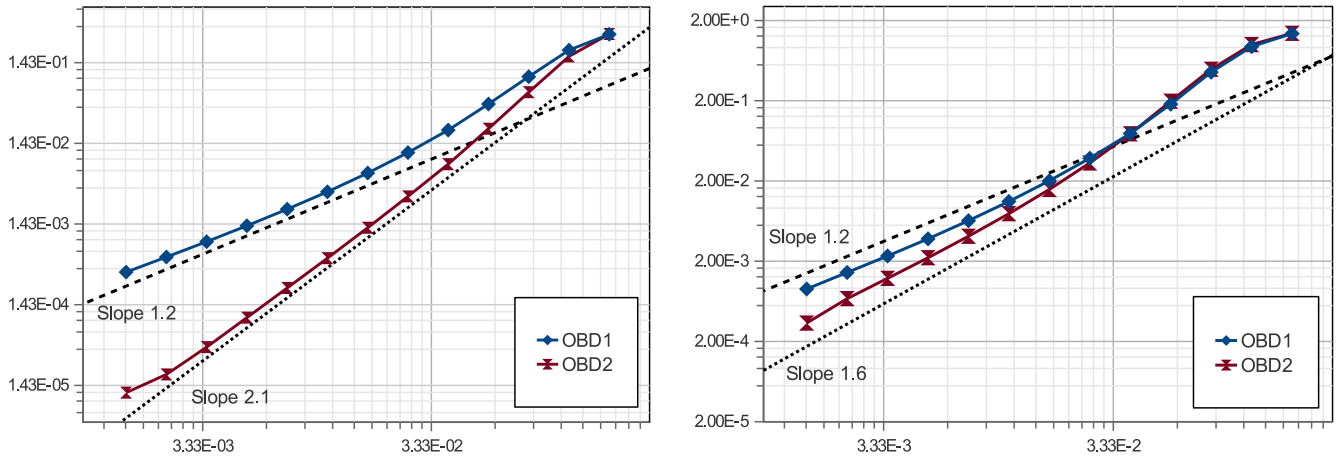


Fig. 3. Time convergence rates with the standard incremental scheme at $t = 2$ with $\Delta x = 1/256$ for the unsteady homogeneous case. Velocity (left) and pressure (right).

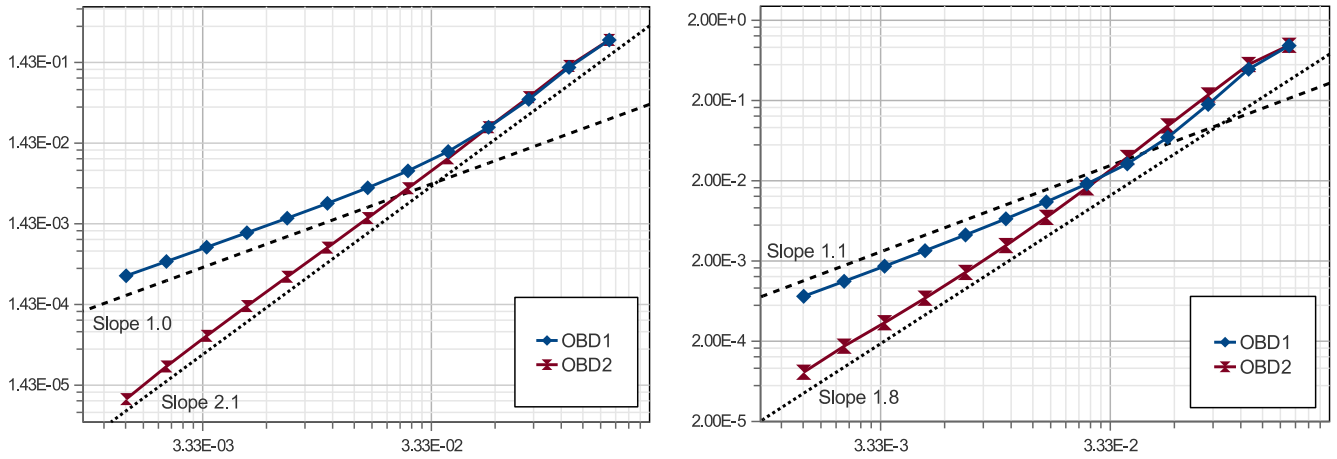


Fig. 4. Time convergence rates with the standard incremental scheme at $t = 2$ with $\Delta x = 1/256$ for the unsteady non-homogeneous case. Velocity (left) and pressure (right).

Remark 1. In Figs. 5 and 6 (right) there is a saturation of the time error by space error at small time step due to space discretisation being too coarse. This behaviour is illustrated in Fig. 7 which represents the time convergence for two meshes. We can see that the finest the mesh is the more saturation is postponed to a smaller time step.

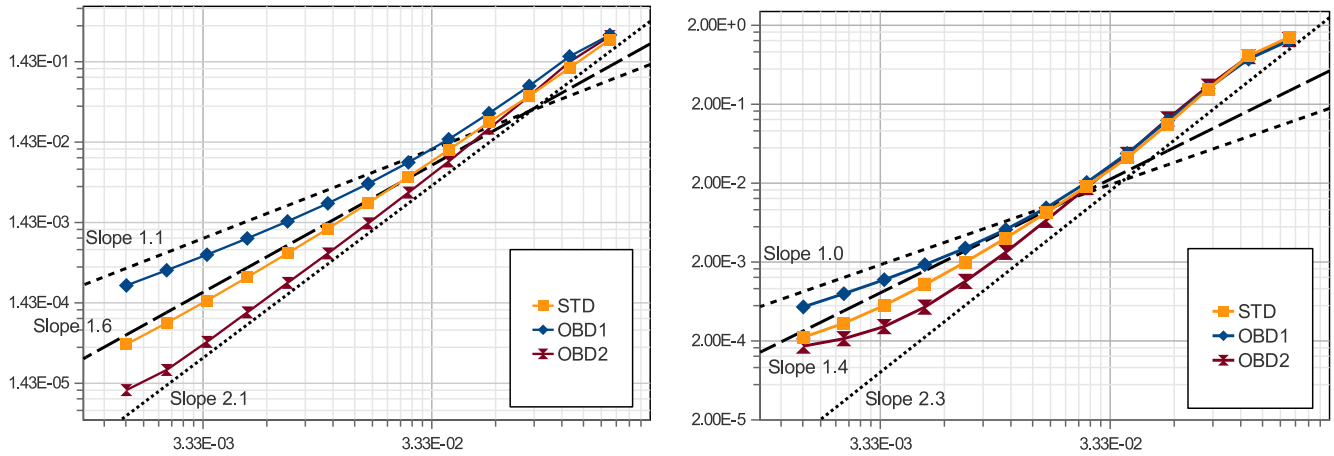


Fig. 5. Time convergence rates with the rotational incremental scheme at $t = 2$ with $\Delta x = 1/256$ for the unsteady homogeneous case. Velocity (left) and pressure (right).

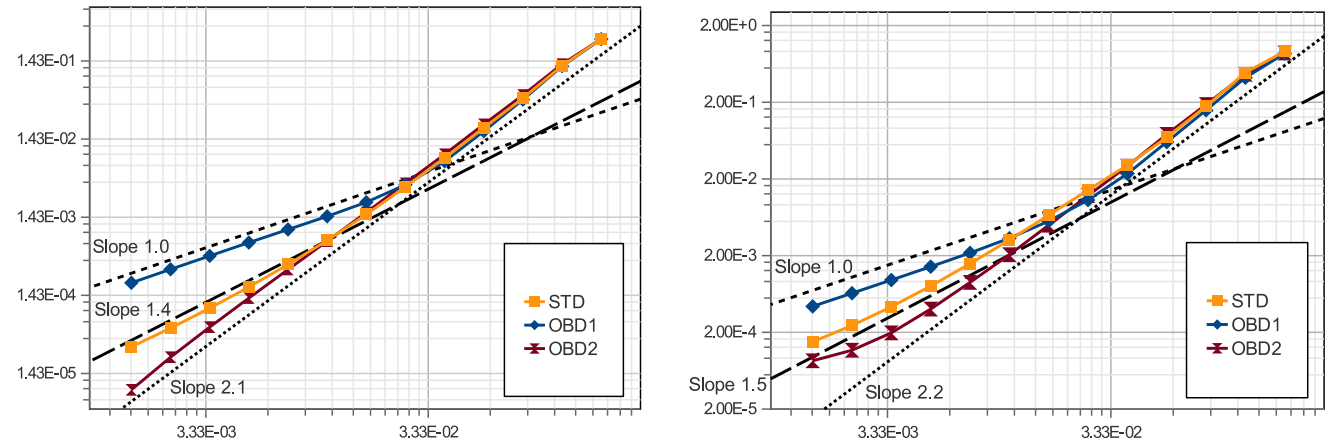


Fig. 6. Time convergence rates with the rotational incremental scheme at $t = 2$ with $\Delta x = 1/256$ for the unsteady non-homogeneous case. Velocity (left) and pressure (right).

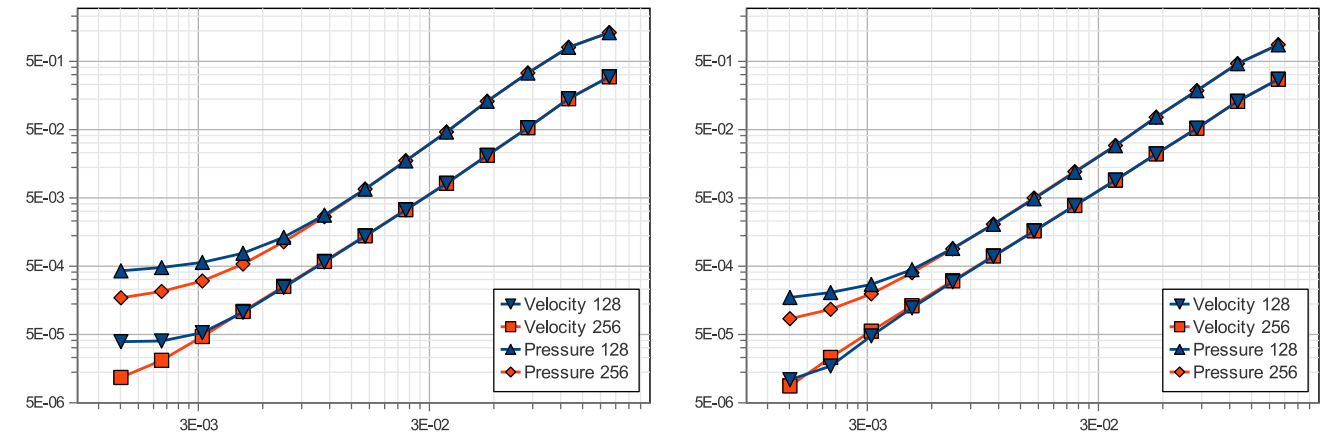


Fig. 7. Time convergence rates for the unsteady homogeneous case (left) and non-homogeneous case (right) with the rotational incremental scheme and OBD2 at $t = 2$ with $\Delta x = 1/128$, $\Delta x = 1/256$.

Remark 2. With Dirichlet boundary conditions on all boundaries, rotational scheme improves error level, compared to standard incremental one, while convergence order remains the same. With the standard traction boundary conditions ($\varphi = 0$ on Γ_N), we can note that rotational scheme improves convergence order. With the proposed method, a more usual effect of the rotational methods which improves only error levels is noted. This can be verified in Figs. 8 and 9.

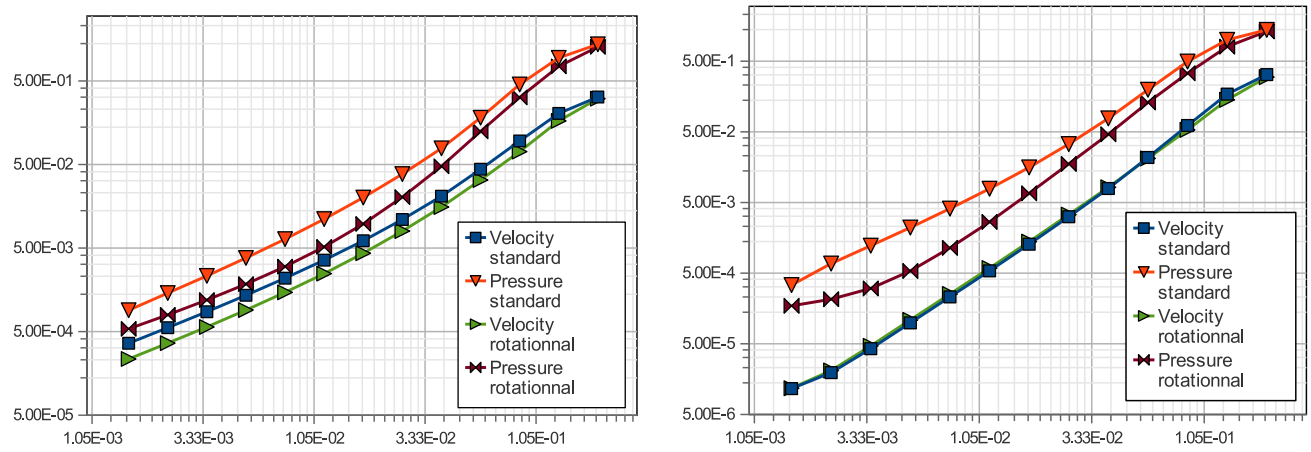


Fig. 8. Comparison between convergence rates with the rotational incremental scheme and the standard incremental scheme at $t = 2$ with $\Delta x = 1/256$ for the unsteady homogeneous case. First order (left) and second order (right) boundary conditions.

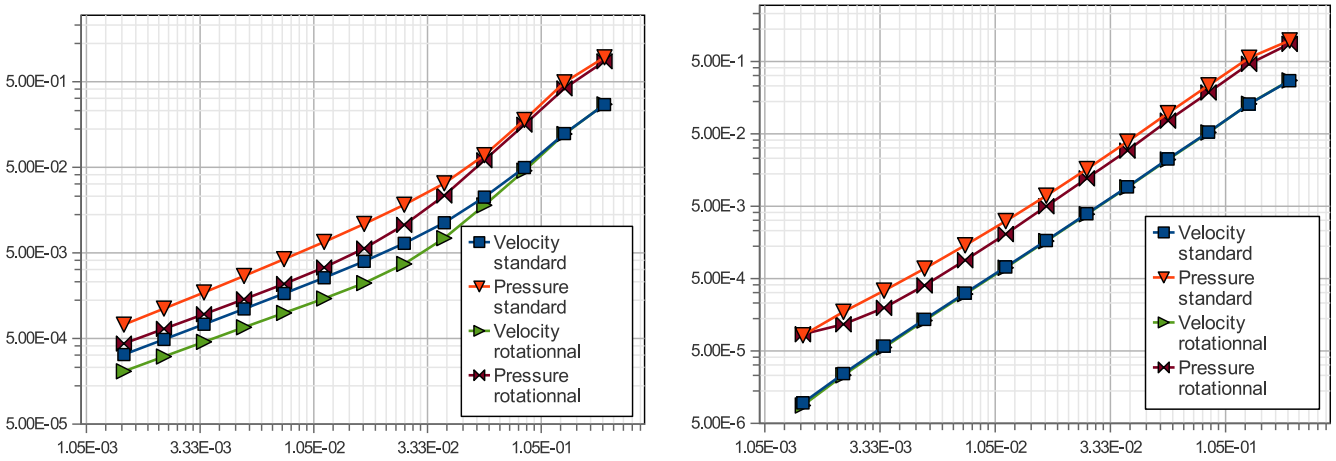


Fig. 9. Comparison between convergence rates with the rotational incremental scheme and the standard incremental scheme at $t = 2$ with $\Delta x = 1/256$ for the unsteady homogeneous case. First order (left) and second order (right) boundary conditions.

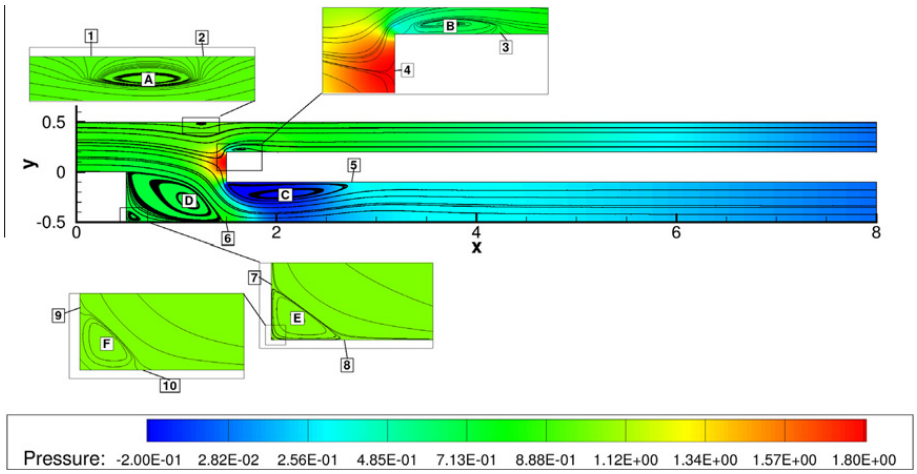


Fig. 10. Steady state of the bifurcation case. Pressure and streamlines. Eddies are numbered with letters.

Remark 3. A similar study was carried out with the open boundary condition (homogeneous or not); the norm $\|\varphi^{\Delta t}\|_{l^2(E)}$ (see (1.1)) was also used. Since the results lead to the same conclusions, they are not shown here.

Table 1
Details of some parameters in the bifurcated tube for different meshes.

Space step size (m)	0.02	0.01	0.005	0.0025	0.00125
Total kinetic energy ($kg \cdot m \cdot s^{-2}$)	$2.0142 \times 10^{+00}$	$2.0025 \times 10^{+00}$	$1.9965 \times 10^{+00}$	$1.9937 \times 10^{+00}$	$1.9923 \times 10^{+00}$
Top outflux ($m^2 \cdot s^{-1}$)	1.9803×10^{-01}	1.9623×10^{-01}	1.9505×10^{-01}	1.9443×10^{-01}	1.9410×10^{-01}
Bottom outflux ($m^2 \cdot s^{-1}$)	3.0117×10^{-01}	3.0357×10^{-01}	3.0490×10^{-01}	3.0556×10^{-01}	3.0589×10^{-01}
Top charge drop ($m^3 \cdot s^{-2}$)	4.2338×10^{-01}	4.1251×10^{-01}	4.0639×10^{-01}	4.0309×10^{-01}	4.0131×10^{-01}
Bottom charge drop ($m^3 \cdot s^{-2}$)	4.2223×10^{-01}	4.1189×10^{-01}	4.0607×10^{-01}	4.0292×10^{-01}	4.0123×10^{-01}

Table 2
Extrapolation of some parameters in the bifurcated tube.

	Extrapolated value	Extrapolation order
Total kinetic energy ($kg \cdot m \cdot s^{-2}$)	$1.9909 \times 10^{+00}$	1.07
Top outflux ($m^2 \cdot s^{-1}$)	1.9374×10^{-01}	0.92
Bottom outflux ($m^2 \cdot s^{-1}$)	3.0623×10^{-01}	1.01
Top charge drop ($m^3 \cdot s^{-2}$)	3.9924×10^{-01}	0.89
Bottom charge drop ($m^3 \cdot s^{-2}$)	3.9925×10^{-01}	0.89

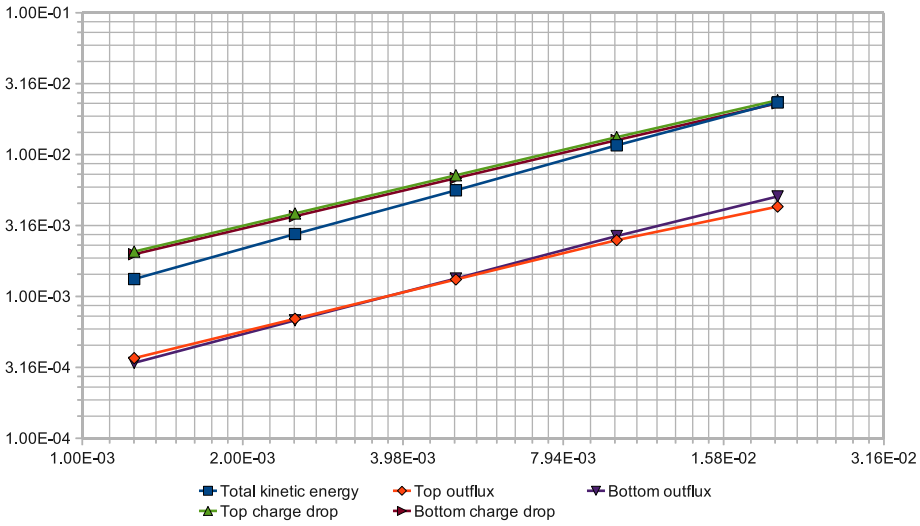


Fig. 11. Spatial convergence rates based on the extrapolation.

As a conclusion on the error convergence rate in time, the proposed method improves orders of the standard incremental scheme from $O(\Delta t)$ to $O(\Delta t^2)$ for the velocity and from $O(\Delta t^{1/2})$ to between $O(\Delta t^{3/2})$ and $O(\Delta t^2)$ for pressure. It also slightly improves the rotational scheme to a clear convergence rate of $O(\Delta t^2)$ for velocity and pressure.

5.3. Numerical results for the Navier–Stokes flows

The boundary conditions proposed are tested with two laminar incompressible physical problems: the steady flow in a bifurcated tube proposed recently by [20] and the unsteady flow past a square section cylinder (see [24,33–36]). The rotational scheme is used with second order time discretisation.

5.3.1. Flow in a bifurcated tube

We deal with the Navier–Stokes equation set on the domain shown in the Fig. 10:

$$\Omega = [0,8] \times [-0.5,0.5] \setminus \{[0,0.5] \times [-0.5,0] \cup [1.5,8] \times [-0.1,0.2]\}.$$

The inflow boundary at $\{x_1 = 0\text{m}\}$ is a Dirichlet one set to a Poiseuille flow with a unitary influx. The two outflow boundaries at $\{x_1 = 8\text{m}\}$ are the proposed open boundary condition OBC2, the remaining boundaries being the no-slip condition. Initialisation is made with $\boldsymbol{u}^0 = \boldsymbol{0}$ and $p^0 = 0$.

Fig. 10 shows steady state for $Re = 600$ based on the height of the larger section ($\rho = 1\ \text{kg m}^{-3}$ and $\nu = 1/600\ \text{m}^2\ \text{s}^{-1}$). The flow is composed by six eddies. An infinite series of Moffatt corner vortices (D, E, F) of increasingly smaller amplitude (see

Table 3
Details of eddies characteristics in the bifurcated tube for different meshes.

Δx(m)	0.02	0.01	0.005	0.0025	0.00125
Eddy A x	$1.2512 \times 10^{+0}$	$1.2534 \times 10^{+0}$	$1.2574 \times 10^{+0}$	$1.2594 \times 10^{+0}$	$1.2606 \times 10^{+0}$
Eddy A y	4.7816×10^{-1}	4.7686×10^{-1}	4.7756×10^{-1}	4.7362×10^{-1}	4.7306×10^{-1}
Eddy A ω	5.6036×10^{-1}	6.0344×10^{-1}	6.8107×10^{-1}	7.0257×10^{-1}	7.1643×10^{-1}
Point 1 x	$1.2000 \times 10^{+0}$	$1.1200 \times 10^{+0}$	$1.0850 \times 10^{+0}$	$1.0675 \times 10^{+0}$	$1.0600 \times 10^{+0}$
Point 2 x	$1.3000 \times 10^{+0}$	$1.3400 \times 10^{+0}$	$1.3600 \times 10^{+0}$	$1.3725 \times 10^{+0}$	$1.3763 \times 10^{+0}$
Eddy B x	$1.6449 \times 10^{+0}$	$1.6258 \times 10^{+0}$	$1.6234 \times 10^{+0}$	$1.6225 \times 10^{+0}$	$1.6219 \times 10^{+0}$
Eddy B y	2.2609×10^{-1}	2.2516×10^{-1}	2.2237×10^{-1}	2.2070×10^{-1}	2.1971×10^{-1}
Eddy B ω	$-8.4218 \times 10^{+0}$	$-1.0078 \times 10^{+1}$	$-9.1216 \times 10^{+0}$	$-8.5552 \times 10^{+0}$	$-8.2187 \times 10^{+0}$
Point 3 x	$1.7600 \times 10^{+0}$	$1.8100 \times 10^{+0}$	$1.8200 \times 10^{+0}$	$1.8175 \times 10^{+0}$	$1.8163 \times 10^{+0}$
Point 4 y	1.0000×10^{-1}	1.0000×10^{-1}	1.0000×10^{-1}	1.0000×10^{-1}	9.8750×10^{-2}
Eddy C x	$2.1183 \times 10^{+0}$	$2.0945 \times 10^{+0}$	$2.0844 \times 10^{+0}$	$2.0803 \times 10^{+0}$	$2.0781 \times 10^{+0}$
Eddy C y	-2.1844×10^{-1}	-2.1645×10^{-1}	-2.1498×10^{-1}	-2.1406×10^{-1}	-2.1336×10^{-1}
Eddy C ω	$1.0540 \times 10^{+1}$	$1.0180 \times 10^{+1}$	$9.9620 \times 10^{+0}$	$9.8574 \times 10^{+0}$	$9.8142 \times 10^{+0}$
Point 5 x	$2.7800 \times 10^{+0}$	$2.7900 \times 10^{+0}$	$2.7950 \times 10^{+0}$	$2.7975 \times 10^{+0}$	$2.7975 \times 10^{+0}$
Eddy D x	$1.1225 \times 10^{+0}$	$1.1340 \times 10^{+0}$	$1.1384 \times 10^{+0}$	$1.1402 \times 10^{+0}$	$1.1410 \times 10^{+0}$
Eddy D y	-2.9328×10^{-1}	-2.9682×10^{-1}	-2.9818×10^{-1}	-2.9874×10^{-1}	-2.9899×10^{-1}
Eddy D ω	$-2.7661 \times 10^{+0}$	$-2.7885 \times 10^{+0}$	$-2.7943 \times 10^{+0}$	$-2.7949 \times 10^{+0}$	$-2.7946 \times 10^{+0}$
Point 6 x	$1.4400 \times 10^{+0}$	$1.4700 \times 10^{+0}$	$1.4700 \times 10^{+0}$	$1.4725 \times 10^{+0}$	$1.4750 \times 10^{+0}$
Eddy E x	5.6857×10^{-1}	5.6466×10^{-1}	5.6229×10^{-1}	5.6094×10^{-1}	5.6023×10^{-1}
Eddy E y	-4.4269×10^{-1}	-4.4493×10^{-1}	-4.4646×10^{-1}	-4.4729×10^{-1}	-4.4723×10^{-1}
Eddy E ω	2.9552×10^{-2}	2.6039×10^{-2}	2.4409×10^{-2}	2.3540×10^{-2}	2.3088×10^{-2}
Point 7 y	-3.8000×10^{-1}	-3.8000×10^{-1}	-3.8000×10^{-1}	-3.8000×10^{-1}	-3.8000×10^{-1}
Point 8 x	6.6000×10^{-1}	6.6000×10^{-1}	6.6500×10^{-1}	6.6500×10^{-1}	6.6375×10^{-1}
Eddy F x	–	–	–	5.0269×10^{-1}	5.0316×10^{-1}
Eddy F y	–	–	–	-4.9732×10^{-1}	-4.9684×10^{-1}
Eddy F ω	–	–	–	-6.5702×10^{-5}	-1.3642×10^{-4}
Point 9 y	–	–	–	-4.9500×10^{-1}	-4.9375×10^{-1}
Point 10 x	–	–	–	5.0500×10^{-1}	5.0625×10^{-1}

[37]) appears in the lower left corner due to the sudden expansion of the section. Three other recirculations (A, B, C) are attached to the horizontal walls due to the contraction of the section. The outflux are equal to 0.194 $m^2\,s^{-1}$ on top and 0.306 $m^2\,s^{-1}$ on bottom, which are qualitatively the same as in [20] where an open boundary condition has also been used.

In order to propose a reference solution, we compute three other parameters characterising the flow:

- The total kinetic energy

$$e_c = \int_\Omega \frac{1}{2}\rho \boldsymbol{u}^2 \, d\boldsymbol{x}.$$

- The outflux

$$Q = \int_R \boldsymbol{u}\cdot\boldsymbol{n} \, dl.$$

- The charge drops

$$\Delta h = \frac{1}{2}\left(\int_L (\boldsymbol{u}\cdot\boldsymbol{n})^2 \, dl - \int_R (\boldsymbol{u}\cdot\boldsymbol{n})^2 \, dl\right) + \frac{1}{\rho}\left(\int_L p \, dl - \int_R p \, dl\right),$$

where L is the left boundary condition, and R the upper or lower right one.

Richardson extrapolation framework [38,39] is used to compute the convergence rates and extrapolated values of these parameters computed on four meshes of step size h_1, h_2, h_3 and h_4 verifying consecutive ratio equal to two. The convergence rate α and the extrapolated value f_{ext} are given by:

$$\alpha = \frac{\ln\left(\frac{f_{h_1} - f_{h_3}}{f_{h_2} - f_{h_4}}\right)}{\ln\left(\frac{h_1}{h_2}\right)}, \tag{5.51}$$

$$f_{\text{ext}} = \frac{\left(\frac{h_3}{h_4}\right)^\alpha f_{h_4} - f_{h_3}}{\left(\frac{h_3}{h_4}\right)^\alpha - 1}. \tag{5.52}$$

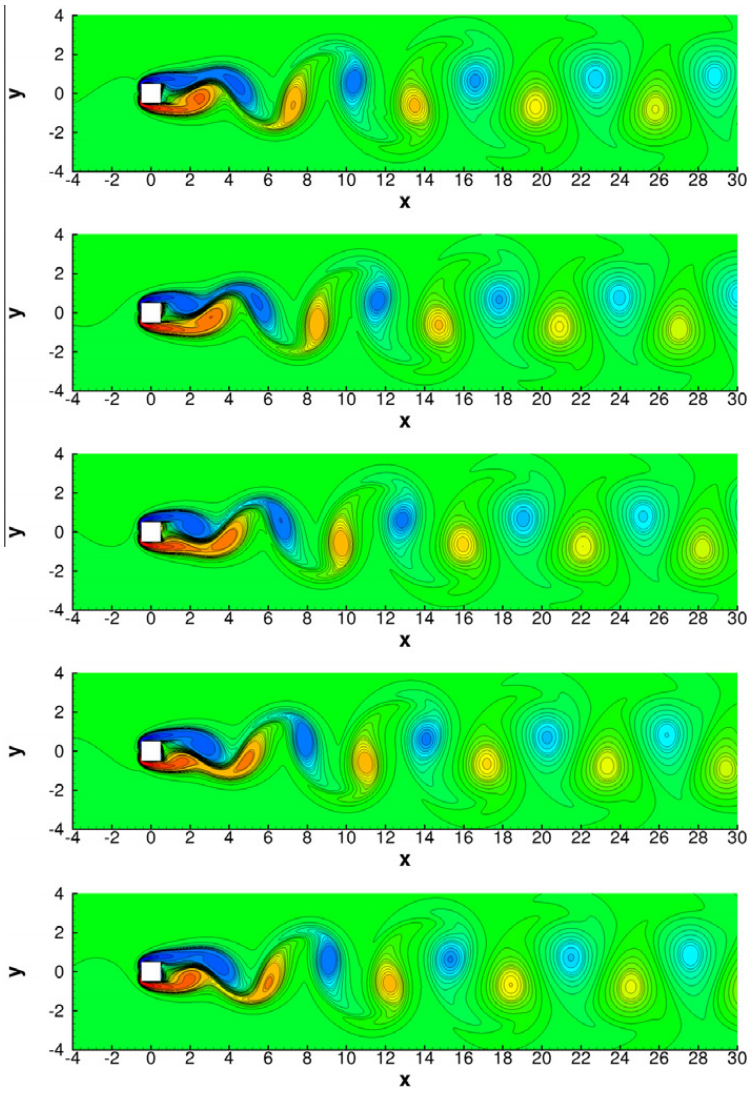


Fig. 12. Vorticity contours during one period.

Table 4
Comparison of computed flow metrics.

References	St	Average C_D	r.m.s C_L
Present study, Ω_1, 6H	0.143235	1.461515	0.153056
Present study, Ω_2, 10H	0.146412	1.474955	0.144166
Present study, Ω_3, 20H	0.147131	1.478540	0.142870
Present study, Ω_4, 30H	0.147167	1.478745	0.142652
Sohankar [34]	0.146	1.46	0.139
Pavlov [36]	0.150	1.51	0.137
Hasan [24]	0.144	1.40	-
Pontaza [33]	0.140	1.48	0.141
Okajima [35] (Exp)	0.143	-	-

Table 5
Time convergence.

Time step (s)	2.4×10^{-2}	1.2×10^{-2}	6.0×10^{-3}	3.0×10^{-3}	Extrapolated value	Extrapolation order
Strouhal number	0.147239	0.147187	0.147170	0.147167	0.147166	1.79
Average drag coefficient	1.479280	1.478870	1.478765	1.478745	1.478739	2.04
r.m.s lift coefficient	0.143079	0.142748	0.142667	0.142652	0.142647	2.10

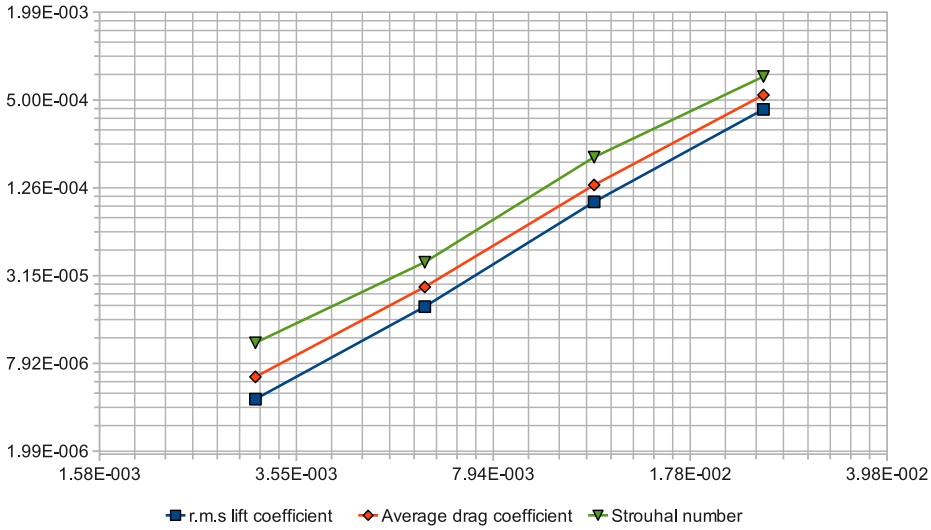


Fig. 13. Time convergence rates based on the extrapolation.

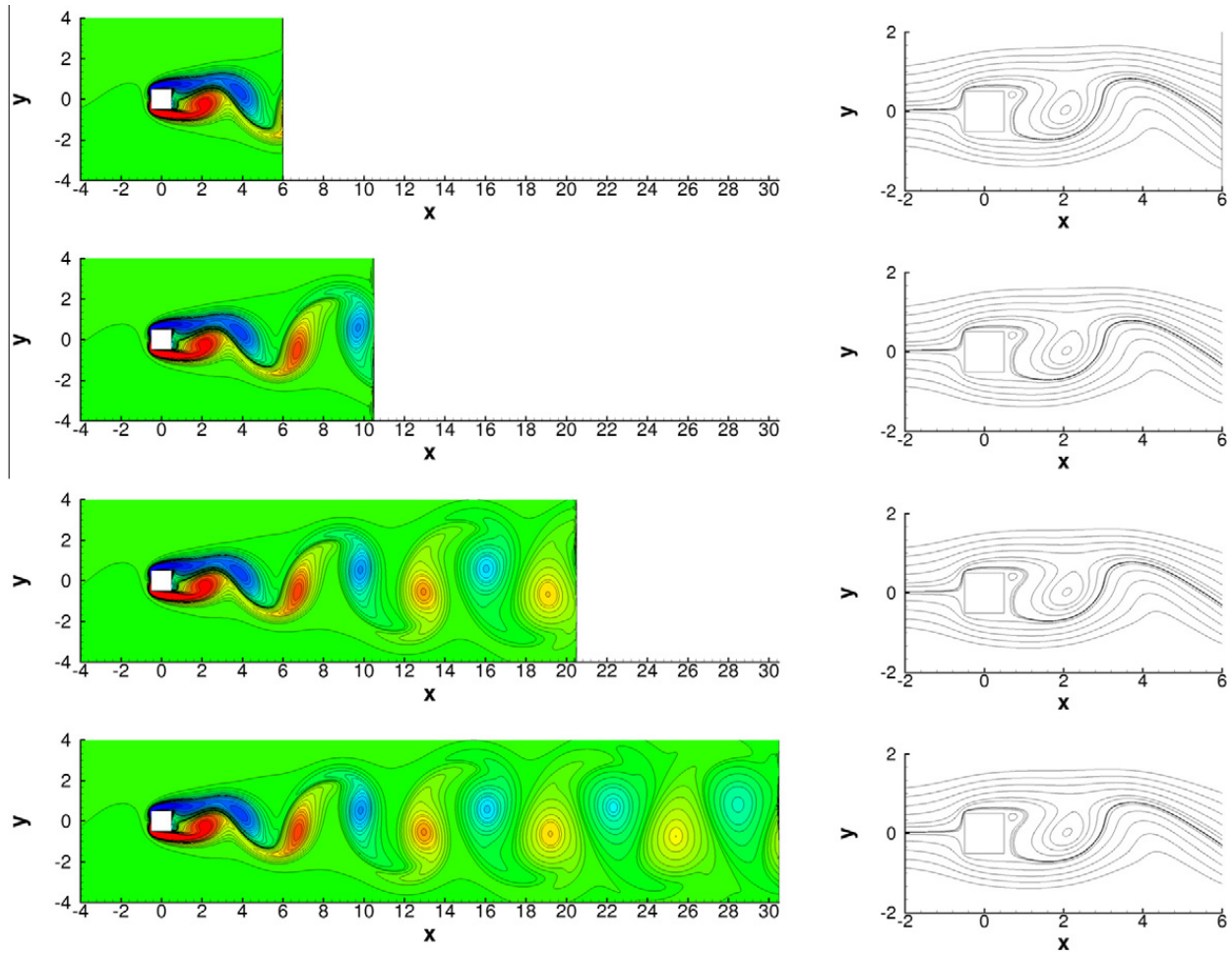


Fig. 14. Instantaneous vorticity contours (left) and streamlines (right) for four meshes, from top to bottom: Ω_1, Ω_2, Ω_3 and Ω_4. In order to compare, isovalues for vorticity contours are the same whatever the domain.

The values for five meshes are detailed in Table 1 and the convergence rates computed with the four finest meshes are in Table 2. A first order space convergence rate is obtained, which is caused, we presume, by the singularity of the geometry at the corners of the domain [40]. We verify on Fig. 11 that results are within the asymptotic convergence zone by plotting

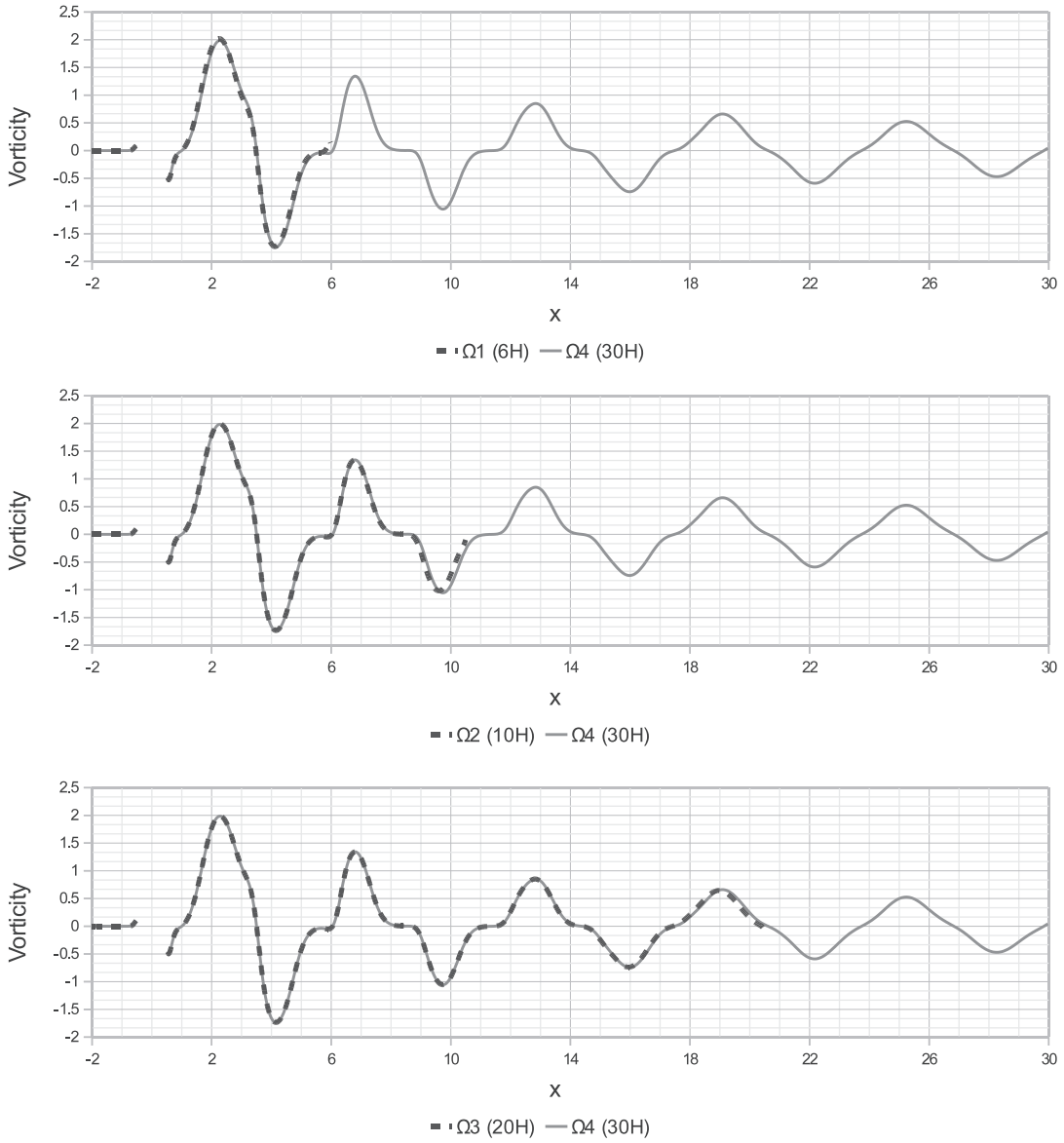


Fig. 15. Comparison of instantaneous vorticity along the line $y = 0$ between Ω_4 and the three other meshes.

the errors against the extrapolated solutions. Finally, we propose in Table 3 a detailed description of the eddies for five meshes: position (x_1, y_2) and vorticity (ω) at the center, and detachment and reattachment points coordinates.

5.3.2. Flow past a square section cylinder

We consider the two dimensional unsteady flow past a square cylinder studied by [34,36,24,33,35] with a normal incidence and $Re = 100$. The Reynolds number is based on the free-stream velocity ($u_\infty = 1\ \mathrm{m\ s^{-1}}$), the square width $H = 1$ m, the density $\rho = 1\ \mathrm{kg\ m^{-3}}$ and the viscosity $\nu = 0.001\ \mathrm{m^2\ s^{-1}}$. We consider four computational domains where the distance between the outflow boundary condition and the cylinder is, respectively, 6H, 10H, 20H and 30H:

- $\Omega_1 = [-10.5, 6.5] \times [-10.5, 10.5]$
- $\Omega_2 = [-10.5, 10.5] \times [-10.5, 10.5]$
- $\Omega_3 = [-10.5, 20.5] \times [-10.5, 10.5]$
- $\Omega_4 = [-10.5, 30.5] \times [-10.5, 10.5]$

The inflow boundary at $\{x_1 = -10.5\ \mathrm{m}\}$ is a Dirichlet condition set to a constant horizontal flow, the outflow boundary at $\{x_1 = 6.5\ \mathrm{m}\}$, $\{x_1 = 10.5\ \mathrm{m}\}$, $\{x_1 = 20.5\ \mathrm{m}\}$ or $\{x_1 = 30.5\ \mathrm{m}\}$ is the proposed open boundary condition OBC2. A symmetry condition is imposed on the upper and lower boundaries. No-slip condition is enforced on the square obstacle placed at $(x_1, x_2) = (0, 0)$. The meshes have around two million points with a constant space step of 0.002 m in the sub-domain $[-2, 4] \times [-2, 2]$. Step size increases toward the boundaries. The initialisation of pressure is made with $p^0 = 0$. To destabilise

the solution, an unsymmetrical velocity field is initialised: for $x_2 > 0$ $\boldsymbol{u}^0 = (1.1, 0)$ m and, for $x_2 < 0$, $\boldsymbol{u}^0 = (0.9, 0)$ m. The computations are made with $\Delta t = 0.003$ s.

The various global parameters characterising the flow are defined as:

- The lift coefficient $C_L = \frac{2F_{x_2}}{\rho u_\infty^2 H}$
- The drag coefficient $C_D = \frac{2F_{x_1}}{\rho u_\infty^2 H}$
- The Strouhal number $St = \frac{fH}{U_\infty}$

where f is the shedding frequency. F_{x_1} and F_{x_2} are the sums of both pressure and viscous forces in the x_1 and x_2 direction around the Γ_c boundary of the cylinder: $(F_{x_1}, F_{x_2}) = \int_{\Gamma_c} \sigma . \boldsymbol{n}$.

The result is a periodic Bénard-von Kármán vortex street shown on the Fig. 12. Values of the Strouhal number, the lift coefficient and the drag coefficient are detailed in Table 4, and are in agreement with the literature.

For the last domain Ω_4, we proceed to a time convergence study. Four time steps are chosen with a consecutive ratio of two. A convergence order around two is measured (see Table 5), in agreement with the formal convergence of the methods. We check on the Fig. 13 that the solutions are within the asymptotic convergence zone by plotting the errors against the extrapolated solutions.

Figs. 14 and 15 show the influence of the size of the computational domain on the rotational and the streamlines at the same phase (first time step just after sign change of the rotationnal at $(x, y) = (1, 0)$ m). As in [24], the position of the outflow boundary condition does not induce distortion of the vortices nor disturb flow around the cylinder. Nevertheless, we can see some distortions near the boundary condition, which reminds us that the open boundary condition is not a "universal" outlet boundary condition. These discrepancies are spatially limited to the boundary as shown on the Fig. 15 which represents vorticity profiles along the line $y = 0$.

References

[1] K.J. Arrow, L. Hurwicz, H. Uzawa, Studies in Linear and Non-Linear Programming, Stanford University Press, Stanford, 1958.
[2] M. Fortin, R. Glowinski, Méthodes de Lagrangien Augmenté – applications à la résolution numérique de problèmes aux limites, 1982.
[3] M.O. Deville, P.F. Fischer, E.H. Mund, High-Order Methods for Incompressible Fluid Flow, Cambridge University Press, Cambridge, 2002.
[4] V. Girault, P. Raviart, Finite Element Methods for Navier-Stokes Equations, Springer Verlag, Berlin, 1986.
[5] A. Chorin, Numerical solution of the Navier–Stokes equations, Mathematics of Computation 22 (1968) 745–762.
[6] R. Témam, Navier Stokes Equations: Theory and Numerical Analysis, North-Holland Pub. Co., Amsterdam, 1984.
[7] 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.
[8] L.J.P. Timmermans, P.D. Minev, F.N. Van De Voss, An approximate projection scheme for incompressible flow using spectral elements, International Journal for Numerical Methods in Fluids 22 (1996) 673–688.
[9] J. Shen, On error estimates of the projection methods for the Navier–Stokes equations: Second-order schemes, Mathematics of Computation 65 (1996) 1039–1066.
[10] J.-L. Guermond, Calculation of incompressible viscous flows by an unconditionally stable projection FEM, Journal of Computational Physics 132 (1997) 12–33.
[11] J.-L. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Computer Methods in Applied Mechanics and Engineering 195 (2006) 6011–6045.
[12] S. Tsynkov, Numerical solution of problems on unbounded domains. A review, Applied Numerical Mathematics 27 (1998) 465–532.
[13] R.L. Sani, P.M. Gresho, Résumé and remarks on the open boundary condition minisymposium, International Journal for Numerical Methods in Fluids 18 (1994) 983–1008.
[14] I. Orlanski, A simple boundary condition for unbounded hyperbolic flows, Journal of Computational Physics 21 (1976) 251–269.
[15] B. Engquist, Absorbing boundary conditions for numerical simulation of waves, Proceedings of the National Academy of Sciences 74 (1977) 1765–1766.
[16] B. Engquist, A. Majda, Numerical radiation boundary conditions for unsteady transonic flow, Journal of Computational Physics 40 (1981) 91–103.
[17] G. Jin, M. Braza, A nonreflecting outlet boundary condition for incompressible unsteady Navier–Stokes calculations, Journal of Computational Physics 107 (1993) 239–253.
[18] M. Forestier, R. Pasquetti, R. Peyret, Spatial development of wakes using a spectral multi-domain method, Applied Numerical 33 (2000) 207–216. C.
[19] M.V. Salvetti, P. Orlandi, R. Verzicco, Numerical simulations of transitional axisymmetric coaxial jets, AIAA Journal 34 (1996) 736–743.
[20] J. Liu, Open and traction boundary conditions for the incompressible NavierStokes equations, Journal of Computational Physics 228 (2009) 7250–7267.
[21] D. Coutand, S. Shkoller, On the interaction between quasilinear elastodynamics and the Navier–Stokes equations, Archive for Rational Mechanics and Analysis 179 (2005) 303–352.
[22] J.-L. Guermond, P. Minev, J. Shen, Error analysis of pressure-correction schemes for the time-dependent stokes equations with open boundary conditions, SIAM Journal on Numerical Analysis 43 (2005) 239–258.
[23] C.-H. Bruneau, P. Fabrie, Effective downstream boundary conditions for incompressible Navier–Stokes equations, International Journal for Numerical Methods in Fluids 19 (1994) 693–705.
[24] N. Hasan, S. Anwer, S. Sanghi, On the outflow boundary condition for external incompressible flows: a new approach, Journal of Computational Physics 206 (2005) 661–683.
[25] C. Février, J. Laminie, P. Poullet, P. Angot, On the penalty-projection method for the NavierStokes equations with the MAC mesh, Journal of Computational and Applied Mathematics 226 (2009) 228–245.
[26] M. Kirkpatrick, S. Armfield, Open boundary conditions in numerical simulations of unsteady incompressible flow, ANZIAM Journal 50 (2008) 760–773.
[27] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Physics of Fluids 8 (1965) 2182–2189.
[28] E. Ahusborde, S. Glockner, A 2D block-structured mesh partitioner for accurate flow simulations on non-rectangular geometries, Computers & Fluids 43 (2011) 2–13.
[29] Hypre, Hypre high performance preconditionner user's manual, <http://acts.nersc.gov/hypre/>, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 2008.

[30] P. Amestoy, Multifrontal parallel distributed symmetric and unsymmetric solvers, Computer Methods in Applied Mechanics and Engineering 184 (2000) 501–520.
[31] S. Schaffer, A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients, SIAM Journal on Scientific Computing 20 (1998) 228–242.
[32] P.N. Brown, R.D. Falgout, J.E. Jones, Semicoarsening multigrid on distributed memory machines, SIAM Journal on Scientific Computing 21 (2000) 1823.
[33] J. Pontaza, J. Reddy, Least-squares finite element formulations for viscous incompressible and compressible fluid flows, Computer Methods in Applied Mechanics and Engineering 195 (2006) 2454–2494.
[34] A. Sohankar, C. Norberg, L. Davidson, Low-Reynolds-number flow around a square cylinder at incidence: study of blockage, onset of vortex shedding and outlet boundary condition, International Journal for Numerical Methods in Fluids 26 (1998) 39–56.
[35] A. Okajima, Strouhal numbers of rectangular cylinders, Journal of Fluid Mechanics 123 (2006) 379–398.
[36] A. Pavlov, S. Sazhin, R. Fedorenko, A conservative finite difference method and its application for the analysis of a transient flow around a square prism, Methods for Heat and Fluid Flow 10 (2000) 6–46.
[37] H.K. Moffatt, Viscous and resistive eddies near a sharp corner, Journal of Fluid Mechanics 18 (2006) 1–18.
[38] J. Richardson, L.F. Gaunt, The deferred approach to the limit. Part I. Single lattice. Part II. Interpenetrating lattices, Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 226 (1927) 299–361.
[39] P.J. Roache, Verification and Validation in Computational Science and Engineering, Hermosa Publishers, Albuquerque, 1998.
[40] M. Dauge, Elliptic boundary value problems in corner domains, Lecture Notes in Mathematics 1341 (1988) 1–257.

6 A fourth-order accurate curvature computation in a level set framework for two-phase flows subjected to surface tension forces

Journal of Computational Physics 305 (2016) 838–876

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A fourth-order accurate curvature computation in a level set framework for two-phase flows subjected to surface tension forces

Mathieu Coquerelle *, Stéphane Glockner

Université de Bordeaux, CNRS, UMR 5295, Bordeaux INP, 16, avenue Pey-Berland, 33607 PESSAC Cedex, France

ARTICLE INFO

Article history:
Received 28 January 2015
Received in revised form 16 September 2015
Accepted 8 November 2015
Available online 10 November 2015

Keywords:
Surface tension
Curvature computation
Level set method
Spurious currents
Two-phase flow
Continuum surface force
Balanced force algorithm
Closest point method

ABSTRACT

We propose an accurate and robust fourth-order curvature extension algorithm in a level set framework for the transport of the interface. The method is based on the Continuum Surface Force approach, and is shown to efficiently calculate surface tension forces for two-phase flows. In this framework, the accuracy of the algorithms mostly relies on the precise computation of the surface curvature which we propose to accomplish using a two-step algorithm: first by computing a reliable fourth-order curvature estimation from the level set function, and second by extending this curvature rigorously in the vicinity of the surface, following the Closest Point principle. The algorithm is easy to implement and to integrate into existing solvers, and can easily be extended to 3D. We propose a detailed analysis of the geometrical and numerical criteria responsible for the appearance of spurious currents, a well known phenomenon observed in various numerical frameworks. We study the effectiveness of this novel numerical method on state-of-the-art test cases showing that the resulting curvature estimate significantly reduces parasitic currents. In addition, the proposed approach converges to fourth-order regarding spatial discretization, which is two orders of magnitude better than algorithms currently available. We also show the necessity for high-order transport methods for the surface by studying the case of the 2D advection of a column at equilibrium thereby proving the robustness of the proposed approach. The algorithm is further validated on more complex test cases such as a rising bubble.

1. Introduction

Surface tension is a key mechanical force in multi-phase flows where it plays a particularly important role in hydrodynamics at small scales. Rising bubbles and airborne water drops are common examples of such flows (see for example [1–3]). The surface tension force lies at the interface between the two phases and is thus singular at that location. This singularity makes it a complicated continuum mechanics problem when coupled with the Navier–Stokes equations.

From a numerical point of view, this singular force is challenging as it requires precise localization of the surface, its associated normal vector and curvature. A variety of methods have been introduced to localize the surface advected by the flow, the front-tracking method [4] in a Lagrangian framework, the Volume Of Fluid (VOF) method [5] and the Level Set (LS) method [6] in a Eulerian framework. These three widely used approaches offer advantages and drawbacks such that these

* Corresponding author.
E-mail addresses: mathieu.coquerelle@bordeaux-inp.fr (M. Coquerelle), glockner@ipb.fr (S. Glockner).

http://dx.doi.org/10.1016/j.jcp.2015.11.014
0021-9991/

methods are actively researched today. We base our original work on the level set framework which permits easy capture of topological changes and offers the advantage of simpler algorithms and a natural extension from 2D to 3D. However, in comparison to the Lagrangian representation, precise localization of the surface is lost and the level set method usually suffers from more volume loss than, for example, VOF methods. A wide range of techniques has been proposed to alleviate these problems such as the hybrid particle level set [7], the CLSVOF method [8] and high-order semi-Lagrangian particles method [9]. As we demonstrate in this article, accurate transport of the surface is required to attain high-order computation of the dependent curvature. In a Eulerian framework, the surface is immersed in the fluid through the use of a Dirac mass and a Heaviside function to characterize the phases.

The surface tension force is commonly introduced in the Navier–Stokes equations as an external force located at the interface. The Continuum Surface Force (CSF) model introduced in [10] gives an elegant solution to regularize the singular term. The method is based on spreading the force over a small neighborhood around the interface. In the last decade, research has been focused on the reduction of spurious currents, arising in the vicinity of the surface [11–16]. These currents are numerical parasitic velocities introduced in the flow by errors in the estimation of the surface tension force. As demonstrated in [17], they are quadratically proportional to the curvature which leads to the need of an accurate curvature computation. Following the observations of François et al. [11], in the CSF model the creation of such currents is due to:

1. non-coherent numerical differentiation in solving the Navier–Stokes equations,
2. non-coherent computations of the surface position, normal vector and curvature.

The first criterion has been recently addressed by the introduction of a balanced-force algorithm [11] which is based on coherent numerical schemes for computing the surface tension force and the associated pressure jump. This technique reduces spurious currents down to machine precision when the curvature is prescribed exactly.

The second cause of these currents is still being investigated today through improving the accuracy of the interface normal and curvature computations, which are direct derivatives of the surface and thus also singularly located. In consequence, they need to be treated adequately in a Eulerian framework. For VOF methods, the Height Function (HF) [18] approach (along with some other improvements) is more precise and drastically reduces spurious currents as shown in [14]. However, their accuracy relies on the precision of the underlying phase function, itself dependent on the VOF transport algorithm (we refer the reader to [19] for a comparative study of the PLIC, WLIC, THINC and CLSVOF methods), which hardly reaches second-order in space accuracy as it is discussed in §2.4.1. More complex methods such as MOF [20] can improve the accuracy of the volume fraction while still not being precise enough to ensure spatial convergence of the surface curvature after its transport. That is why recent research focuses on coupled methods like CLSVOF [8] or CLSMOF [21] which use the underlying level set for more accuracy on the curvature computation.

For level sets, a curvature extension based on a closest point algorithm [22] has been efficiently used in [12], for unstructured grids, which reduces spurious currents much more than previous techniques. The implementation of this method is quite simple and can be easily extended to 3D.

In the CSF approach, the accuracy of the curvature calculation is essential to reduce the introduction of errors in the flow. In this paper we present a detailed study of the difficulties arising in the numerical computation of surface tension forces in a Eulerian framework, which relies on the extension of the curvature in the vicinity of the interface. In section 2.4 we develop three key criteria for evaluating the quality of this extension, and illustrate their respective effects on flow dynamics. These criteria are guidelines for the techniques described thereafter. In particular, we assert that a minimized variation of the curvature in the normal direction of the surface is crucial to reduce numerical errors on the surface tension term.

In §3.4, we present the principles of curvature extension for level sets and compare three different methods for their computation, the last being the novel method that we propose:

1. the osculatory circle approximation which consists in a linear extension of the curvature in the normal direction,
2. the Closest Point method based on the interpolation on the surface of a curvature approximation, as proposed by Herrmann in [12],
3. the proposed approach based on an improved Closest Point algorithm that ensures the colinearity criterion, and uses reinterpolation to reduce variation in the normal direction.

We discuss the respective advantages and drawbacks of each technique in order to propose a general understanding of curvature extension in a level set framework. The accuracy of the methods is then compared and analyzed on geometric cases in §4.2. All three methods give very good results with circular geometry. In order to discriminate between the methods, we studied the case of the ellipse which is more challenging because the variation of the curvature is less trivial. Consequently, the proposed Closest Point algorithm with colinearity properties and reinterpolation is finally retained for its high accuracy and robustness.

We first validate our approach for the classical static column, first with constant density §4.3 and then with variable density §4.4, and prove its ability to drastically reduce spurious currents at fourth-order convergence rate. Popinet [14] studied the spurious currents when a column is advected by a constant velocity. This case is a good challenge for surface tension methods because it induces more errors at the interface. In §4.5, we report very satisfactory results demonstrating

Table 1
Terminology. In the paper, all units are given in the International System of Units.

Symbol	Definition	Symbol	Definition
ρ	Fluid density	Γ	A surface
μ	Fluid dynamic viscosity	$\mathbf{n}$	Surface normal
$\mathbf{u}$	Fluid velocity	τ	Surface tangent
p	Pressure	κ	Surface curvature
σ	Surface tension coefficient	ϕ, ψ	Level set functions
H	The Heaviside function	$\widetilde{\kappa}$	Discrete representation for any κ
δ	The Dirac mass	κ_{ex}	Exact value for any κ
h	The spatial discretization step	κ_σ	The standard deviation of the curvature
Δt	The temporal discretization step	$\kappa_{\sigma,\mathbf{n}}$	The normal standard deviation of the curvature
$\mathbf{x}_{ij}$	The discrete value of $\mathbf{x}$ at i, j	RMS	Root Mean Square

that the algorithm proposed is robust to the transport of the surface. This robustness is due to high-order resolution of the advection equation coupled with accurate curvature extension. Finally, we validate our algorithm with a comparison of numerical results obtained from the simulation of the zero gravity drop oscillation §4.6 and the 2D bubble rise §4.7. This demonstrates that the approach proposed can be effectively used in Navier Stokes solvers based on level set methods.

2. Governing equations

2.1. Terminology

Before presenting the equations we present in Table 1 the terminology and notations that will be used through the whole document.

2.2. Two-phase fluid flow equations in the presence of surface tension forces

The incompressible Navier–Stokes equations with variable densities and viscosities following the continuum–surface–force (CSF) approach originally proposed by Brackbill et al. [10] can be written as:

$$\begin{aligned} \rho(\partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}) &= -\nabla p + 2\nabla\cdot(\mu\mathbf{D}) + \sigma\kappa\delta_\Gamma\mathbf{n} \\ \partial_t\rho + (\mathbf{u}\cdot\nabla)\rho &= 0 \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned} \tag{1}$$

with $\mathbf{D} = \left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)/2$ the deformation tensor and p the pressure field. The term $\sigma\kappa\delta_\Gamma\mathbf{n}$ is singularly located on the surface Γ separating the two phases by a Dirac delta function δ_Γ, oriented in the normal direction $\mathbf{n}$, with curvature κ and a physical surface tension coefficient σ. Equation (1) is valid under the assumption a constant σ along the surface between the two phases which is a common assumption in most fluid dynamics computations.

2.3. Level set representation for two-phase flows

In the level set approach, the phases of the fluid are spatially identified in n dimensions using a function $\phi : \mathbb{R}^n \to \mathbb{R}$ such that each phase stands on one side of a certain level set value α. Thus, they are separated by the surface:

$$\Gamma(t) = \{\mathbf{x} \,|\, \phi(\mathbf{x}, t) = \alpha\}. \tag{2}$$

For the sake of simplicity we choose $\alpha = 0$, implying that the first (resp. second) phase stands where ϕ is negative (resp. positive).

The local concentration c_i – or characteristic function – of the ith phase is written

$$c_i(\mathbf{x}, t) = H_i(\phi(\mathbf{x}, t)),$$

where $H_1(y) = H(y) = 1 - H_2(y)$ and $H(y)$ is the Heaviside function defined as $H'(y) = \delta(y)$.

The density ρ is expressed over the whole domain as a function of ϕ:

$$\rho(\mathbf{x}, t) = H(\phi(\mathbf{x}, t))\rho_1 + (1 - H(\phi(\mathbf{x}, t)))\rho_2, \tag{3}$$

where ρ_i is the constant density of the ith phase. A similar equation represents the viscosity $\mu(\mathbf{x}, t)$.

As in common level set methods, the function ϕ is passively transported by the fluid through the advection equation:

$$\partial_t\phi + \mathbf{u}\cdot\nabla\phi = 0 \tag{4}$$

which implicitly follows the position of the surface over time.

The normal and curvature fields can be computed for the whole domain as:

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}, \tag{5}$$

and

$$\kappa = \nabla\cdot\mathbf{n} = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \tag{6}$$

Equations (5) and (6) coincide with the normal to and the curvature of the surface Γ for points $\mathbf{x}$ such as $\phi(\mathbf{x}) = \alpha$.

Due to the numerical discretization in the level set Eulerian framework we work with a smooth representation of those quantities where δ_Γ and H are not singular but spread in the vicinity of Γ. Thus we will use the term **extension** for any quantity that physically stands on the surface but is numerically and spatially extended around it.

2.4. Problems arising with surface tension forces

As stated in the introduction, a qualitative computation of surface tension forces to minimize the spurious currents is based on two criteria:

Criterion 1. *The consistent numerical discretization of the pressure and surface tension terms.*

Criterion 2. *The rational computation of the surface's curvature.*

By rational computation we mean a curvature computation that assures the three following aspects:

Criterion 2.1. *Fast convergence to the exact curvature.*

Criterion 2.2. *Minimized deviation along the surface.*

Criterion 2.3. *Minimized variation along the surface's normal.*

Several efficient and accurate numerical methods have been developed over the last decade focusing on the computation of the curvature and the surface tension term. The balanced-force algorithm used in different flow simulation frameworks and based on the Ghost Fluid Method [11], VOF [14], level set on unstructured grids [12] has greatly reduced the numerical errors by satisfying Criterion 1 with methods suitable for the curvature computation to satisfy Criterion 2.

Our flow solver presented in section 3 is based on the same balanced-force principle and an improved curvature computation that complies with Criteria 2.1, 2.2 and 2.3.

For the sake of clarity we explain the importance of those 3 criteria using a 2D case of the static column equilibrium which has been widely used in the literature as a minimal and representative case.

2.4.1. Fast convergence to the exact curvature

The first criterion is essential to computationally converge to the physical experiments, as shown in Fig. 2(a). Even in a stable simulation, a significant error in the calculation of the curvature can drastically change the hydrodynamics. This becomes particularly important when the computational grid is refined as high curvatures can arise and often dominate the dynamic processes at those scales.

Hence it is essential to have a precise and fast convergent computation of the curvature, at least at the same order as the flow solver. Whatever the choice Eulerian or Lagrangian for the surface representation, the curvature depends on the second derivatives of a given function ϕ. For example, the curvature of a parameterized curve $\Gamma = (x(t), y(t))$ is:

$$\kappa = \frac{x'y'' - y'x''}{\left(x'^2 + y'^2\right)^{3/2}}$$

or given by equation 6 for level sets.

In this section, we use the term "error" in a general context. In a numerical point of view, we use the most restrictive L_∞ norm to define the error E_∞ between an analytical solution ψ and its numerical approximation ϕ as:

$$E_\infty(\psi,\phi) = \max|\psi - \phi|\,.$$

Errors in second derivative computation.

Proposition 1. *The error in the second derivative is of order $m-2$ when the error on the surface representation is of order m.*

To prove this, we first illustrate the impact of numerical errors on the second derivatives of a function. This demonstration can be extended to the general case of surface and curvature computation, regardless of whether the surface is Eulerian or Lagrangian.

Let ψ be a 1D function of x, sufficiently differentiable. In addition, let ϕ be a perturbed version of ψ such that $\phi(x,h,m) = \psi(x) + e(x,h,m)$, where $e(x,h,m)$ represents a perturbation of amplitude of order h^m with spatial frequency $1/h$ (where, in a numerical computation, h is the spatial discretization, i.e. the smallest variation that can be captured). The associated numerical error is:

$$E_\infty(\psi,\phi) = \max|\psi(x) - \phi(x,h,m)| = \max|e(x,h,m)| = O(h^m).$$

The nth derivative of ϕ is a function of the nth derivatives of ψ and $e(x,h,m)$:

$$\frac{\partial^n \phi(x,h,m)}{\partial x^n} = \frac{\partial^n \psi(x)}{\partial x^n} + \frac{\partial^n e(x,h,m)}{\partial x^n}.$$

Then the perturbation on the nth derivative of ϕ will be of the order h^{m-n}, i.e. the associated numerical error is:

$$E_\infty\left(\frac{\partial^n \psi}{\partial x^n}, \frac{\partial^n \phi}{\partial x^n}\right) = \max\left|\frac{\partial^n \psi(x)}{\partial x^n} - \frac{\partial^n \psi(x,h,m)}{\partial x^n}\right| = \max\left|\frac{\partial^n e(x,h,m)}{\partial x^n}\right| = O(h^{m-n}).$$

Hence, if $m > n$ then the function $\frac{\partial^n \phi}{\partial x^n}$ will converge toward $\frac{\partial^n \psi}{\partial x^n}$ at a positive rate of $O(h^{m-n})$, otherwise it will not converge and lead to errors $> O(1)$ increasing as $h \to 0$. This is completely independent of the accuracy of the underlying numerical scheme used to discretize the quantities, especially the curvature.

A detailed example is given in Appendix A.

Errors in the initialization and transport of the surface. When a function ϕ represents the surface, whether in a Lagrangian or Eulerian fashion, and if the error in its initialization and/or transport is less than the order 2, then the errors in the second derivatives of ϕ – hence the curvature – will be at least $O(1)$, given the previous demonstration. This will lead to increasing error in the flow as the spatial discretization tends to 0.

Therefore, in order to have at least a second-order error in the computation of the curvature, it is mandatory to **initialize** and **transport** the surface with errors less than order h^4 compared to the exact solution. This is true regardless of the errors in the velocity field used to transport the surface as the exact solution also has to be transported by the perturbed flow.

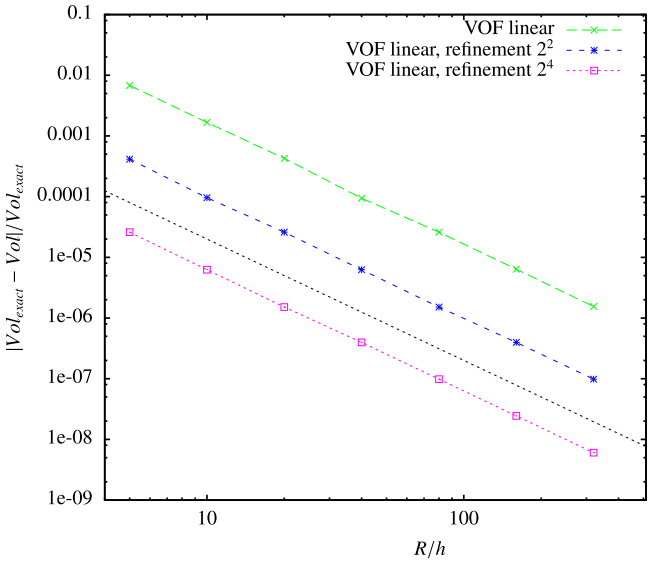
Examples for second-order precision

- In a Lagrangian representation, the discretized points have to stand closer than h^4 to the exact surface. The velocity interpolation at those points must be at least fourth-order.
- In a level set/signed distance approach, the distance to the surface has to be computed at least to fourth-order, which is easily attainable for common geometries. The transport equation must be solved at the same order, as well as reinitialization algorithms to ensure a similar order of precision. If fourth-order methods are used to transport and reinitialize ϕ then the curvature computation will never be better than second-order.
- In VOF methods, the fraction of volume occupied by the phase has to be computed at least to fourth-order, which is not trivially obtainable even for common geometries. In [18] the authors use a refinement technique to initialize the volume fraction more precisely, up to second-order and the curvature is computed with the HF method. In Fig. 9 and in the text of section 5.1, they show a second-order convergence of the curvature error. This statement is not correct as proven in the above paragraph. To demonstrate this we did the exercise with the same parameters and methods (with the HF algorithm implemented from [23]). As shown in Fig. 1, the curvature error clearly diverges when using finer meshes than the one used in [18]: what they have observed is a bias visible at coarse meshes (for $R/h < 40$) which is due to reduced numerical errors on the initial volume fraction, relatively to the exact value. However, these errors become dominant with finer discretization (for $R/h \geq 40$) and exhibit non-convergence, as proven in Proposition 1. More precise methods for the initialization have been proposed since, like for example [24], but as VOF is not the representation chosen for our method, we will not discuss the question further. Additionally, the transport equation also has to be at least fourth-order. Fig. 1 also shows that, even with the exact volume fraction computation, the use of a low order transport technique does not lead to spatial convergence for the curvature. Until recently, no VOF transport method was sufficiently precise to attain high-order dynamic computation of the curvature; recent methods such as DRAC [25] are fourth-order accurate in space.

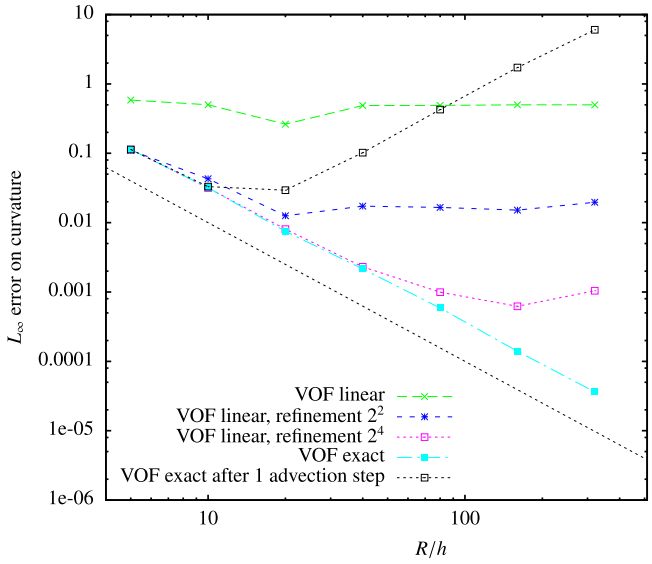
2.4.2. Minimized deviation along the surface

In the particular case of a static column in 2D (resp. a static sphere in 3D), the overall deviation of the curvature along the surface can be expressed through the standard deviation. It is described on the surface as:

$$\tilde{\kappa}_\sigma = \sqrt{\int_\Gamma (\tilde{\kappa} - \tilde{\kappa}_{mean})^2} \tag{7}$$



(a) Convergence of the L_1 error on the volume of the disk.



(b) Non convergence of the L_∞ curvature error for the low-order initialization and after transport of the exact volume fractions.

Fig. 1. L_1 volume and L_∞ curvature error with VOF representation, for a disk of radius $R = 0.25$ centered in a box of size 1, $h = 1/N$ is the spatial discretization step. The volume fractions are initialized with linear segments, with or without mesh refinement (like proposed in [18]) or with the exact solution obtained by integration of the circle equation. The curvature is computed with the second-order accurate Height Function algorithm implemented from [23].

where the tilde denotes discrete measures, the σ symbol is used here to mean standard deviation and $\tilde{\kappa}_{mean}$ is the numerically computed mean curvature:

$$\tilde{\kappa}_{mean} = \int_\Gamma \tilde{\kappa}. \tag{8}$$

As stated in [14], this second criterion is numerically relevant as, in the case of a balanced force algorithm, it reduces the spurious currents to the machine precision. This statement is true even in the case of a poor computation of the absolute physical curvature, i.e. even when $|\tilde{\kappa}_{mean} - \kappa_{exact}|$ is high.

For the static column equilibrium problem, the presence of oscillations on the curvature evaluation along the interface will result in the appearance of capillary waves, thus creating undesired spurious currents in the whole simulation. It is important to note that this criterion is hardly extendable to the general case where the curvature is not constant along the surface. However, one can picture those small variations in the curvature as a bias in the geometry between the position of the surface, its normal vector and curvature that will disrupt the flow dynamic through the surface tension errors, as it is depicted in Fig. 2(b).

In a more general context where the curvature is not constant, the deviation along the surface is less trivial. It can be expressed in the proximity of $\mathbf{x}$ a point on the surface Γ, as:

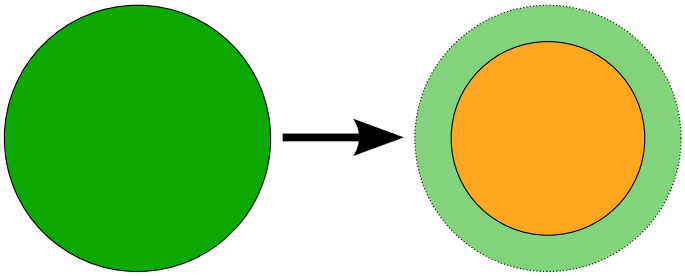
$$\tilde{\kappa}_\sigma(\mathbf{x}) = \sqrt{\int_{-\epsilon}^{\epsilon} \left(\tilde{\kappa}\left(\Gamma_{\mathbf{x}}(s)\right) - \tilde{\kappa}_{exact}\left(\Gamma_{\mathbf{x}}(s)\right)\right)^2 ds}$$

where $\Gamma_{\mathbf{x}}(s)$ is a parameterization of the surface around $\mathbf{x}$, with $\Gamma_{\mathbf{x}}(0) = \mathbf{x}$, and with $\epsilon \to 0$.

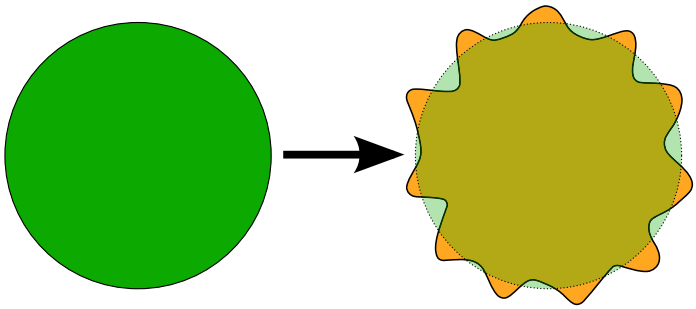
2.4.3. Minimized variation along the surface's normal

Following the description in §2.3 where the surface is implicitly represented in a Eulerian level set framework, a common strategy to solve PDEs on the phases or surfaces is to smoothly extend the physical properties in a region around the interface. Hence the third criterion, which can be geometrically understood as orthogonal to the previous one, consists in minimizing the errors in the calculation of the extension of the curvature, which will be treated in detail in §3.4.4. The precise extrapolation of quantities in the normal direction is necessary in other applications, for example for the mean curvature motion in a Eulerian framework (see [26] for a level set study). In a VOF framework, a smoothing or an extension of the curvature is computed in order to obtain a better estimation of κ at grid nodes, as presented in [18,27] for example.

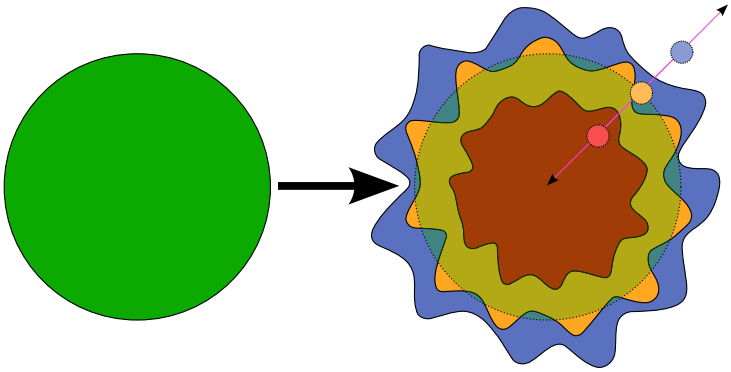
Errors in the normal direction will deviate the dynamics of the flow by introducing forces in the vicinity of the surface that will create undesired perturbations, as depicted in Fig. 2(c). Hence, it is necessary to compute an accurate curvature extension, i.e. a curvature that is constant in the normal direction, as pointed out by [17].



(a) Representation of the dynamic effect of curvature's absolute error. Here the curvature is overestimated hence the dynamic of the surface will be similar to the one of the smaller orange circle.



(b) Representation of the dynamic effect of curvature's variations along the surface. Even though the surface's position may be exact (the translucid green circle with dotted surface), the error on the curvature will mislead the dynamic as if the geometry was erroneous (the virtual orange perturbed surface).



(c) Representation of the dynamic effect of curvature's variations along the surface's normal. Even though the surface's position may be exact (the translucid green circle with dotted surface), the error on the extended curvature in the normal direction will mislead the dynamics as if the geometry was erroneous. The red, orange and blue perturbed forms correspond respectively to curvature computed on three different level sets, for example $\phi = \{-h, 0, +h\}$. The normal direction is noted with a purple segment on which three circles have been placed to represent spatial points at a distance of $\{-h, 0, +h\}$ from the exact surface.

Fig. 2. Visual representation of the effects of the different errors on curvature following Criteria 2.1, 2.2, 2.3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the case of a level set representation of the surface, it can be easily seen that a simple computation of the curvature as $\kappa = \nabla \cdot \mathbf{n}$ will lead to a high deviation in the normal direction, as depicted in Fig. 3(a) representing the level sets of a circle. Moreover, we can see in this example that the curvature will be higher inside the circle and lower outside, compared to the curvature on the surface. This fact is true on any smooth geometry as the spatial discretization tends to zero.

In the presence of surface tension, this last phenomenon will dynamically stretch the level sets on one side of the surface while compacting it on the other side. This can lead to more numerical errors as the flow moves the level sets, particularly when using a signed distance function that will be highly distorted, requiring frequent reinitializations of the level set function.

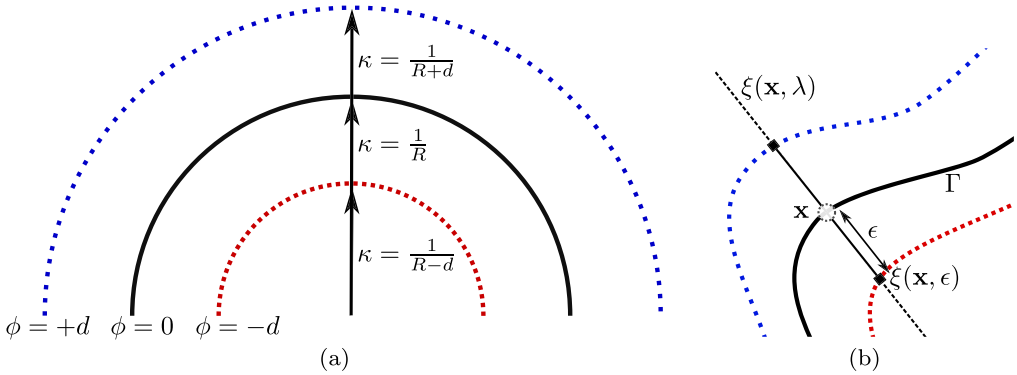


Fig. 3. (a) Local evaluation of the curvature based on a signed distance level set function. The curvature varies as $1/(R-\phi)$ on the ϕ's level set, R being the exact radius of the osculatory circle of the surface at $\phi = 0$. (b) Graphical representation of the line $\xi(\mathbf{x}, \lambda) = \mathbf{x} + \lambda \mathbf{n}(\mathbf{x})$ passing through $\mathbf{x}$ and following the normal $\mathbf{n}(\mathbf{x})$ of the surface Γ.

Given that the surface tension forces are spread in the vicinity of the surface, if the variation of the curvature along the normal is null then the surrounding level sets will move at the same speed hence reducing the numerical errors. In the case of the column equilibrium problem, this will reduce the spurious currents.

As a consequence, in order to reduce parasitic currents, one has to minimize the deviation $\kappa_{\sigma,\mathbf{n}}$ of the curvature along the normal. It can be expressed similarly to eq. (7) at a point $\mathbf{x}$ on the surface Γ, given any curvature κ that is extended in the whole domain:

$$\tilde{\kappa}_{\sigma,\mathbf{n}}(\mathbf{x}) = \sqrt{\int_{-\epsilon}^{\epsilon} \left(\tilde{\kappa}\left(\xi(\mathbf{x}, \lambda)\right) - \tilde{\kappa}(\mathbf{x})\right)^2 d\xi}$$

where $\xi(\mathbf{x}, \lambda) = \mathbf{x} + \lambda \mathbf{n}(\mathbf{x})$ is the line passing through $\mathbf{x}$ and following the normal $\mathbf{n}(\mathbf{x})$, as illustrated in Fig. 3(b). This measurement is meaningful in the vicinity of Γ and therefore on the segment $\{\xi(\mathbf{x}, \lambda),\ \lambda \in [-\epsilon, \epsilon]\}$, with $\epsilon \to 0$. In a Eulerian framework, it is evaluated for discrete points $\mathbf{x}$ close to the surface.

2.4.4. Conclusion

In the next section, we present a novel method based on:

1. the balanced force method as developed in [11] and detailed for a level set framework in §3.3,
2. an adaptation of the Closest Point method for curvature extension as first used in [12], that we improved as detailed in §3.4.

Our approach is focused on the accurate and efficient numerical computation of the curvature in the vicinity of the surface complying with fast convergence to the exact curvature (Criterion 2.1) and minimized deviation along the surface (Criterion 2.2) and along the normal direction (Criterion 2.3).

3. Numerical methods

In this section we first describe the general Navier–Stokes flow solver we used, then the level set transport equation algorithm. In the third subsection we present our implementation of the balanced-force surface tension method in a level set framework. We finally present the method that we developed for the accurate computation and extension of the surface curvature. The validation of the method with associated numerical results is presented in section 4 on pure geometric and dynamic cases.

3.1. Flow solver and discretization

For our numerical simulation, we used the Thetis flow solver [28,29]. Time discretization of the momentum equation is a first-order Eulerian scheme with an implicit formulation for the viscous term. The velocity/pressure coupling under the incompressible flow constraint is solved with the time splitting pressure correction method [30]. The equations are discretized on a staggered grid by means of the finite volume method. The space derivatives of the inertial and stress terms are discretized by a second-order centered scheme. Since the phase function is not defined on each grid point where viscosities and densities are needed for the Navier–Stokes discretization, the physical characteristics are interpolated on the staggered grid. We use a linear interpolation to calculate the density on the velocity nodes, whereas a harmonic interpolation is used for the viscosity. Linear systems of the prediction and correction steps are solved thanks to the direct MUMPS solver [31,32].

3.2. Level set transport

In order to obtain an accurate computation of the curvature, as stated in section 2.4.1, the level set function is advanced in time using a 5th order in space WENO method as first described in [33]. A second-order in time Runge Kutta scheme is used based on the advection of ϕ for half a time step:

$$\phi^{n+\frac{1}{2}} = \phi^n - \Delta t\,(\mathbf{u}^n \cdot \nabla)\phi^n$$
$$\phi^{n+1} = \phi^n - \Delta t\,(\mathbf{u}^n \cdot \nabla)\phi^{n+\frac{1}{2}},$$

with $\phi^n(\mathbf{x}) = \phi(\mathbf{x}, t_n)$ and $\phi^{n+1}(\mathbf{x}) = \phi(\mathbf{x}, t_n + \Delta t)$.

This method limits the spatial errors for ϕ around $O(h^5)$ which, with a precise curvature computation, theoretically yields a third order error in the curvature computation. We will see and explain in section 4 that we reach errors at even higher order in the presence of small velocities.

3.3. Balanced force surface tension

In order to satisfy the consistent numerical discretization scheme criterion (cf. §2.4, Criterion 1), the surface tension term is discretized on the MAC grid at the velocity nodes, i.e. on the faces of the control volume, with the same finite difference scheme used for the Poisson pressure equation. As in typical balanced force algorithms, we use the CSF approximation for the surface tension term rewritten relative to the phase concentration c (we dropped the index for clarity, posing $c = c_1$):

$$\sigma\kappa\delta_\Gamma\mathbf{n} \equiv \sigma\kappa\nabla c,$$

where ∇c is numerically computed at the faces of the cells, and points from phase 2 to phase 1.

As a consequence, recalling that the concentration c changes its value from 0 to 1 on either side of the interface, ∇c is located on the faces of the cells surrounding Γ depending on the stencil used to compute ∇c and on the smoothness of c.

Remark. Using a first order accurate function for the concentration, i.e. c equals 0 (resp. 1), if ϕ is negative (resp. positive) will lead to an aliased function following the cells' faces. Consequently the pressure profile will directly follow this aliasing as well as the surface. As a cell's level set value can vary with transport from a certain small value $\pm\epsilon$ around zero, the surface tension force (and thus the pressure jump) will undesirably oscillate from one cell to another. Moreover, derivating this function to compute ∇c for the surface tension force will lead to $O(1)$ errors.

Hence it is necessary to use sufficiently accurate representations of the physical quantities in order to avoid the aliasing problems.

A common choice for calculating the concentration is to use a regularized Heaviside function such as:

$$c(\mathbf{x}) = H_\epsilon(\phi(\mathbf{x}))$$

where $\epsilon = O(h)$. This smooth approximation can induce a smoother dynamic behavior. We did not find this to be a problem in the cases we studied in section 4. Common strategies for computing the Heaviside function in a level set framework do not show strict spatial convergence in 2D and 3D when $\epsilon = mh$, as proven in [34] for the associated δ_ϵ function. Moreover, as proven in [35] (and as studied in their references), in presence of high density ratios, the use of a smooth Heaviside function to define the physical quantities is crucial to increase numerical stability. From a physical point of view, the interface is thus perceived as a volume of fixed width assuring a continuous transition between the two phases. For the sake of simplicity, we used the same smoothed Heaviside function to define ρ, μ and c.

Hence, we did not find necessary to implement a more precise scheme, as for example proposed in [36]. The method proposed in this article is totally independent of this choice, however, a precise numerical approximation of the Heaviside function will benefit the overall numerical results.

In order to guarantee a precise enough Heaviside function for sufficiently fine spatial discretizations (as used in our numerical results of section 4), we chose a value of $\epsilon = 2h$ and:

$$H_\epsilon(y) = \begin{cases} 0 & \text{if } y \le -\epsilon \\ 1 & \text{if } y \ge \epsilon \\ \dfrac{1 + y/\epsilon + \sin\left(\frac{y\pi}{\epsilon}\right)/\pi}{2} & \text{otherwise} \end{cases} \tag{9}$$

with its associated smooth Dirac mass

$$\delta_\epsilon(y) = \begin{cases} 0 & \text{if } |y| \le \epsilon \\ \dfrac{1 + \cos\left(\frac{y\pi}{\epsilon}\right)}{2\epsilon} & \text{otherwise.} \end{cases} \tag{10}$$

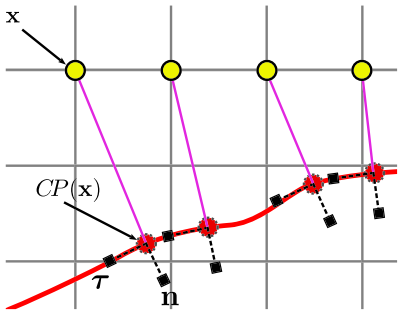


Fig. 4. Closest point method: a geometrical representation. Yellow dots are grid points with their corresponding closest point on the surface as red dots. Normals and tangents on the surface are drawn as dotted square ended segments. The purple lines show the colinearity between $\overrightarrow{\mathbf{x}CP(\mathbf{x})}$ and $\mathbf{n}(CP(\mathbf{x}))$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Moreover, so as to be sure to reduce the effect of interface spreading or shrinking when the level set is not a distance function, we used a renormalized representation of the level set as proposed in [37]:

$$c(\mathbf{x}) = H_\epsilon(\phi(\mathbf{x})/|\nabla\phi(\mathbf{x})|).$$

For the discretization of the ∇c, we used the same second-order scheme as for the pressure gradient: $\partial c_{i+\frac{1}{2},j}/\partial x = (c_{i+1,j} - c_{i,j})/\Delta x$. Hence, with $\epsilon = 2h$, the surface tension force is null everywhere but on the faces in the $3 \times 2 = 6$ cells band surrounding Γ.

Consequently, κ has to be computed accurately only at the center of cells around which ∇c is not null. In our flow solver, the curvature has to be known at the center of the faces of the staggered grid. We discuss this point in §3.4.5.

3.4. Curvature extension in a level set framework

In order to satisfy the rational computation of the surface's curvature criterion (cf. §2.4), one must use numerical methods that prove convergence in space discretization (Criterion 2.1) and minimal variation on the surface (Criterion 2.2) and along its normal (Criterion 2.3).

We present here a novel method to attain this goal in 2 steps:

1. The accurate initial estimation of the curvature,
2. The precise extension of the curvature in the vicinity of the interface.

First, we describe the general principle of the Closest Point method on which our approach is based. Then, we introduce error criteria that will help to analyze the numerical behavior of the method. Finally, we detail the novelties of our method which calculates an accurate curvature extension in a level set framework.

3.4.1. The Closest Point method

The Closest Point (CP) method, first introduced in [38], has been successfully used in [22] to solve PDEs defined on a surface by extending the physical quantities in the vicinity of the surface and then solving the equations in a Eulerian domain. The CSF principle from [10] is based on a parallel point of view for integrating the singular force residing on the surface.

Herrmann [12] first used the CP principle to extend the curvature in a level set framework. It consists in defining the curvature at a point $\mathbf{x}$ as the curvature of its closest point on the interface:

$$\kappa(\mathbf{x}) = \kappa(CP(\mathbf{x})) \tag{11}$$

In a Eulerian framework, it requires the use of interpolation functions as $CP(\mathbf{x})$ will generally be localized between grid points. That principle can be applied to any physical quantity defined on the surface for which an extension is needed. Our approach follows Herrmann's work by improving the CP research algorithm and enhancing the curvature's extension smoothness.

$CP(\mathbf{x})$ is the closest point from $\mathbf{x}$ to the surface Γ and is defined as:

$$CP(\mathbf{x}) = \min_{\mathbf{y}\in\Gamma}(\|\mathbf{x} - \mathbf{y}\|). \tag{12}$$

By construction the vector $\overrightarrow{\mathbf{x}CP(\mathbf{x})}$ is orthogonal to the local tangent at $CP(\mathbf{x})$ and hence is collinear with the normal of the surface at $CP(\mathbf{x})$:

$$\overrightarrow{\mathbf{x}CP(\mathbf{x})} \cdot \boldsymbol{\tau}(CP(\mathbf{x})) = 0 \tag{13}$$

where $\boldsymbol{\tau}$ is the unit tangent orthogonal to $\mathbf{n}$ as shown in Fig. 4.

3.4.2. Error measures

In order to analyze the accuracy of the numerical methods used to compute the curvature, we define four error measurements.

Exact curvature. In order to bound the errors of the numerically computed curvature, we need to define an exact curvature as reference. The curvature is only defined on the surface, in a level set framework, as we are searching for an extended representation of the curvature of a surface. Following equation (11), we define the exact curvature at a point $\mathbf{x}$ as:

$$\kappa_{ex}(\mathbf{x}) = \kappa_{ex}(CP(\mathbf{x})). \tag{14}$$

In the case of a circle of radius R, the exact curvature is constant over the whole domain: $\kappa_{ex}(\mathbf{x}) = 1/R$.

To understand how the closest point method extends physical quantities in the whole domain, we recall from §2.4.3 the definition of the line $\xi(\mathbf{y}, \lambda)$ that passes by the point $\mathbf{y}$ on the surface and follows the normal at $\mathbf{y}$:

$$\xi(\mathbf{y}, \lambda) = \mathbf{y} + \lambda \mathbf{n}(\mathbf{y}) \tag{15}$$

with $\lambda \in \mathbb{R}$. Based on equation (14), we know that $\mathbf{x}$ stands on the line $\xi(CP(\mathbf{x}), \lambda)$.

We can then define the dual extended curvature lines for every point $\mathbf{y}$ of the surface, along their associated normal by:

$$\kappa_{ex,\mathbf{n}}(\xi(\mathbf{y}, \lambda)) = \kappa_{ex}(\mathbf{y}). \tag{16}$$

The collection of segments of extended curvature lines $\kappa_{ex,\mathbf{n}}$ for which $|\lambda|$ is minimal is the extended curvature field κ_{ex} as defined by (14).

It is important to note that ambiguities arise when two or more lines $\xi(\mathbf{y}, \lambda)$ intersect for a same parameter length $|\lambda|$, i.e. when a point $\mathbf{x}$ has two or more closest points on the surface. These cases arise for points relatively far from the surface or when the surface curvature is large. In a numerical framework, it is statistically highly unlikely to encounter the case of two equidistant closest points. However, the closest point to $\mathbf{x}$ will vary from one point on the surface to another when the surface is transported, phenomenon that could lead to temporal jumps in the curvature field. This problem is not assessed in this paper, however we did not encounter any numerical instabilities as the extended curvature field is computed close enough to the surface and the curvature is not too high, i.e. $\kappa \ll h^{-1}$.

In practice, we only require κ_{ex} in the vicinity of a few cells (here N cells) around the interface as required by the CSF approximation (cf. §3.3). For that purpose we introduce the function $\overline{\delta_\Gamma^N}$ defined by:

$$\overline{\delta_\Gamma^N}(\mathbf{x}_{ij}) = \begin{cases} 1 & \text{if } \overline{\delta_\Gamma^{N-1}}(\mathbf{x}_{i'j'}) = 1 \text{ or is next} \\ & \quad \text{to a cell } i'j' \text{ such that } \overline{\delta_\Gamma^{N-1}}(\mathbf{x}_{i'j'}) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

This band around the interface is constructed iteratively based on the cells containing the interface $\overline{\delta_\Gamma^0}$:

$$\overline{\delta_\Gamma^0}(\mathbf{x}_{ij}) = \begin{cases} 1 & \text{if } 0 < c(\mathbf{x}_{ij}) < 1, \\ 0 & \text{otherwise.} \end{cases}$$

In the next sections, for the sake of clarity, we drop the exponent N in $\overline{\delta_\Gamma^N}$.

L_2 and L_∞ norm errors. We define the L_2 and L_∞ normalized errors for the curvature on the overall domain as:

$$L_2 = \frac{1}{\kappa_{ex}} \sqrt{\frac{\sum (\tilde{\kappa} - \kappa_{ex})^2 \overline{\delta_\Gamma}}{\sum \overline{\delta_\Gamma}}}, \tag{18}$$

$$L_\infty = \frac{1}{\kappa_{ex}} \max |(\tilde{\kappa} - \kappa_{ex}) \overline{\delta_\Gamma}|. \tag{19}$$

Standard deviation on the surface. The (normalized) standard deviation on the surface is only significant for a circle/column where the exact curvature is constant along the surface, hence it can only be studied in that case:

$$\tilde{\kappa}_\sigma = \frac{1}{\tilde{\kappa}_{mean}} \sqrt{\sum (\tilde{\kappa} - \tilde{\kappa}_{mean})^2 \overline{\delta_\Gamma}} \tag{20}$$

where

$$\tilde{\kappa}_{mean} = \frac{\sum \tilde{\kappa} \overline{\delta_\Gamma}}{\sum \overline{\delta_\Gamma}}. \tag{21}$$

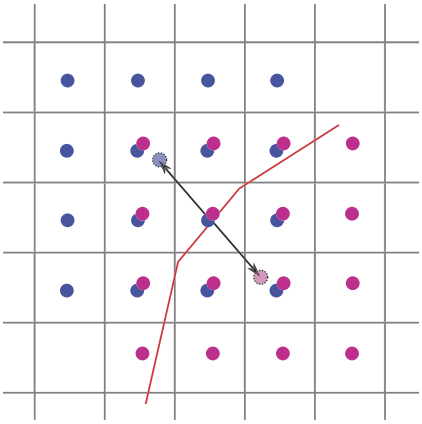


Fig. 5. Stencil used for computing the standard deviation along the normal with fourth-order Lagrange interpolation functions. The surface is shown in red, blue (resp. purple) dots are used for $k=1$ (resp. $k=-1$) in eq. (23) and are spatially shifted for clarity. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Standard deviation along the normal. The (normalized) standard deviation along the line $\xi(\mathbf{y},\lambda)$ defined at point $\mathbf{y}$ is computed as:

$$\kappa_{\sigma,\mathbf{n}}(\mathbf{y}) = \frac{1}{\kappa(\mathbf{y})}\sqrt{\int (\kappa(\xi(\mathbf{y},\lambda)) - \kappa(\mathbf{y}))^2\, d\lambda} \tag{22}$$

which is discretized in the M cells neighborhood of the cell $\mathbf{y}_{i,j}$ as:

$$\tilde{\kappa}_{\sigma,\mathbf{n}}(\mathbf{y}_{i,j}) = \frac{1}{\tilde{\kappa}(\mathbf{y}_{i,j})}\sqrt{\frac{1}{2M+1}\sum_{k=-M}^{+M}(\tilde{\kappa}(\xi(\mathbf{y}_{i,j},kh)) - \tilde{\kappa}(\mathbf{y}_{i,j}))^2}. \tag{23}$$

In practice, as we are looking at the variation of κ in a small area around the interface, we fix $M=1$. It should be noted that fourth-order Lagrange interpolation functions are used to compute $\tilde{\kappa}(\xi(\mathbf{y}_{i,j},kh))$ because $\xi(\mathbf{y}_{i,j},kh)$ will generally not coincide with the mesh points. The stencil used to compute $\tilde{\kappa}_{\sigma,\mathbf{n}}$ around one cell is presented in Fig. 5.

The overall L_2 and L_∞ errors for standard deviation along the normal are defined as:

$$L_{2,\tilde{\kappa}_{\sigma,\mathbf{n}}} = \sqrt{\frac{\sum \overline{\delta_\Gamma}\tilde{\kappa}^2_{\sigma,\mathbf{n}}}{\sum \overline{\delta_\Gamma}}}, \tag{24}$$

$$L_{\infty,\tilde{\kappa}_{\sigma,\mathbf{n}}} = \max \overline{\delta_\Gamma}\tilde{\kappa}_{\sigma,\mathbf{n}}. \tag{25}$$

3.4.3. Accurate estimation of the curvature

Before applying the Closest Point algorithm, we first need to get a precise initial estimation of the curvature at the surface points. Recalling eq. (6), the curvature can be computed in the Eulerian level set framework on the overall domain by numerically computing:

$$\kappa_{LS} = \nabla\cdot\mathbf{n} = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \tag{26}$$

where we use the index *LS* to further differentiate this scalar field with the other methods. In order to reduce the numerical errors due to the computation of higher-order derivatives, we use a formula for the curvature where the second derivative errors are limited to only a part of the calculation:

$$\kappa_{LS} = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) = \frac{1}{|\nabla\phi|}\left(\Delta\phi - \left(\left[D^2\phi\right]\frac{\nabla\phi}{|\nabla\phi|}\right)\cdot\frac{\nabla\phi}{|\nabla\phi|}\right) \tag{27}$$

where $D^2\phi$ is the Hessian matrix of ϕ: $\left[D^2\phi\right]_{i,j} = \frac{\partial^2\phi}{\partial x_i \partial x_j}$.

Understanding the curvature field in a level set framework.

Fact 2. *For any smooth level set function ϕ, as $h\to 0$, the curvature computed by equation (26) on the surface, i.e. $\mathbf{x}\in\Gamma$, is the exact curvature of the surface at $\mathbf{x}$.*

We propose to extend this property to all the surfaces captured by ϕ, i.e. all iso-values:

Claim 3. *For any function ϕ, as $h \to 0$, the curvature computed by equation (26) at a point $\mathbf{x}$ is the exact curvature at $\mathbf{x}$ of the surface defined by the level set $\phi(\mathbf{x}) = \alpha$.*

Those two properties will be used below to express a first approximation of the extension of the curvature in the whole domain. To illustrate this point, under the assumption $\Gamma = \{\mathbf{x} \,|\, \phi(\mathbf{x}) = 0\}$, we describe three cases:

1. If ϕ is a signed distance function, i.e. $\phi(\mathbf{x}) = sd(\mathbf{x})$, then $\kappa_{LS}(\phi, \mathbf{x})$ is the curvature of the surface represented by the level set $\Gamma_{\phi(\mathbf{x})} = \{\mathbf{y} \,|\, \phi(\mathbf{y}) = \phi(\mathbf{x})\}$. The curvature in the whole domain can thus be seen as a stretched/shrunk extension of the curvature at the surface, growing as a function of $sd(\mathbf{x})$, as depicted in Fig. 3.
2. If ϕ can be defined as the composition of another level set ψ with a scalar function f, i.e. $\phi = f(\psi)$, then we can express the curvature by the relationship:
$$\kappa_{LS}(\phi, \cdot) = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) = \nabla \cdot \left(\frac{\nabla f(\psi)}{|\nabla f(\psi)|} \right) = sign\left(f'(\psi)\right) \left(\frac{\nabla \psi}{|\nabla \psi|} \right) = sign\left(f'(\psi)\right) \kappa_{LS}(\psi, \cdot)$$
which gives $\kappa_{LS}(\phi, \cdot) = \kappa_{LS}(\psi, \cdot)$ when f' is strictly positive. Hence any strictly increasing function f applied to a signed distance brings us back to the first case.
3. Otherwise, outside of Γ where ϕ cannot be interpreted as a function of a signed distance function, κ_{LS} may not be directly compared to the surface's curvature. However, if ϕ is sufficiently smooth, a re-normalized form (see [37]) of the level set function $\phi/|\nabla\phi|$ gives a first order approximation of a signed distance function near the interface which brings us back to the first case.

3.4.4. Extension of the surface's curvature

Here we present two different methods for extending the surface curvature and discuss the advantages and drawbacks of each method. The first one is based on Claim 3 and is presented to give a better geometrical understanding of the curvature in a level set framework. The second is the method we propose based on the Closest Point method presented in §3.4.1 and following Herrmann's works [12].

Osculatory circle approximation. This approximation is based on the fact that, under the assumption that locally, as the discretization tends to zero, the curvature of a surface can be defined as the inverse of the radius of the osculatory circle tangent to Γ at $\mathbf{y}$:

$$\kappa(\mathbf{y}) = 1/R(\mathbf{y}).$$

This assertion is also true in a level set framework as shown in Fig. 3(a). Let $d(\mathbf{x})$ be the signed physical distance from the point $\mathbf{x}$ to the surface at level set $\phi_0 = 0$. As $\mathbf{x}$ stands on the line $\mathbf{x} = \xi(\mathbf{y}, \lambda)$ (where ξ is defined by eq. (15)), or conversely $\mathbf{y} = CP(\mathbf{x}) \in \Gamma$, we can write $d(\mathbf{x}) = sign(\lambda).\|\mathbf{x} - \mathbf{y}\|$. Then the radius of the osculatory circle at $\mathbf{x}$ is $R(\mathbf{x}) = R(\mathbf{y}) + d(\mathbf{x}) = \kappa^{-1}(\mathbf{y}) + d(\mathbf{x}) = \kappa^{-1}(\mathbf{x})$.

Consequently we can define an extended surface curvature as:

$$\kappa_{osc}(\mathbf{x}) = \frac{1}{R(\mathbf{y})} = \frac{1}{\kappa^{-1}(\mathbf{x}) - d(\mathbf{x})}$$

which does not require any knowledge of $\mathbf{y}$. In a general level set framework the equation is:

$$\kappa_{osc}(\mathbf{x}) = \frac{1}{\kappa_{LS}^{-1}(\mathbf{x}) - d_\phi(\mathbf{x})} \tag{28}$$

with d_ϕ being the physical signed distance function computed from ϕ which, near the surface, as the discretization step tends to zero, can be approximated by $d_\phi(\mathbf{x}) = \frac{\phi(\mathbf{x})}{|\nabla\phi(\mathbf{x})|}$. In the particular case of a signed distance function $d_\phi(\mathbf{x}) = \phi(\mathbf{x})$.

Limits The limits of this approximation are when:

- the local curvature is null, $\kappa(\mathbf{x}) = 0$, then the osculatory circle has an infinite radius which can raise numerical problems. A threshold is thus used and we set $\kappa_{osc}(\mathbf{x}) = 0$ in those cases,
- the radius $R(\mathbf{x}) = R(\mathbf{y}) + d(\mathbf{x}) = 0$, i.e. $\mathbf{x}$ is the singular center of the osculatory circle, where we do not extend the curvature. As we only look in the vicinity of the surface, this case will only arise for very high curvatures (resp. small radii), $\kappa = O(1/h)$ for which we propose a threshold for the maximum curvature value $\kappa_{max} = 1/h$, which is coherent with the physical maximum curvature that can be captured at the scale h.

This osculatory circle extension is of low order as it does not take into account the variation of a curvature which is essential in the general case. We present in 4.2 spatial convergence results that show this approximation to be fourth-order in the circular case but only first order in the ellipsoidal case.

Closest point on surface interpolation. As introduced in §3.4.1, another approach to extend the curvature away from the surface consists in finding the closest point to $\mathbf{x}$ on Γ and interpolating the curvature there. We present a detailed version of the Closest Point search algorithm that will serve as a base for the novel method proposed hereafter.

Without an analytical representation of the surface, finding $\mathbf{y} = CP(\mathbf{x})$ is not as straightforward as finding on which line $\mathbf{x} = \xi(\mathbf{y}, \lambda)$ stands. However, we can get an approximation of the closest point to $\mathbf{x}$ by following the back trajectory $\mathbf{y} = \xi^{-1}(\mathbf{x}, \lambda)$ starting at $\mathbf{x}$ and leading to $\mathbf{y}$.

In a level set framework, $\xi^{-1}(\mathbf{x}, \lambda)$ follows the gradient field of ϕ and, in general, is a curve and not a line. Thus, one needs to descend along that curve to find $\hat{\mathbf{y}} = \widetilde{CP}(\mathbf{x})$, a numerical approximation of the exact solution $\mathbf{y} = CP(\mathbf{x})$.

We introduce Algorithm 1, noted $CP_{\odot}$, to compute an approximate closest point. $CP_{\odot}$ is based on a first-order Newtonian method with a convergence criterion based on a threshold distance $\epsilon \ll h$ and a with virtual time step Δ_λ. We have used here the local approximation of the distance from the point $\hat{\mathbf{y}}^n$ to the surface: $d_\phi(\hat{\mathbf{y}}^n) = \frac{\phi(\hat{\mathbf{y}}^n)}{|\nabla\phi(\hat{\mathbf{y}}^n)|}$ and of the normal: $\mathbf{n}(\hat{\mathbf{y}}^n) = \frac{\nabla\phi(\hat{\mathbf{y}}^n)}{|\nabla\phi(\hat{\mathbf{y}}^n)|}$. Interpolation is used in order to compute those values from the discrete grid points. Fig. 6 shows a schematic view of the algorithm.

Algorithm 1 $CP_{\odot}$: first order Newton method for the closest point search in a level set framework.

1. Initialize $\hat{\mathbf{y}}^0 = \mathbf{x}$, $i = 0$
2. While $|\phi(\hat{\mathbf{y}}^i)| > \epsilon$ do
   (a) $\hat{\mathbf{y}}^{i+1} = \hat{\mathbf{y}}^i - \Delta_\lambda d(\hat{\mathbf{y}}^i)\mathbf{n}(\hat{\mathbf{y}}^i)$
   (b) $i \leftarrow i + 1$
3. $\widetilde{CP}_{\odot}(\mathbf{x}) = \hat{\mathbf{y}}^i$

In practice, the threshold distance ϵ is set to h^4 as we expect fourth-order convergence on the CP algorithms. The virtual time step is set to $\Delta_\lambda = 0.9$ in order to stay on the same "side" of the level set as much as possible, i.e. not crossing the zero level set in the presence of numerical errors. Setting a lower value for Δ_λ will help to converge more precisely for points far from the surface. However as we are mostly interested by points in the proximity of Γ, we have found this value to be sufficient.

Once the closest point $\widetilde{CP}(\mathbf{x})$ has been computed we interpolate the curvature at that position leading to the following formula for the extended curvature at the grid point $\mathbf{x}_{i,j}$:

$$\tilde{\kappa}_{CP.}(\mathbf{x}_{i,j}) = Interpolate(\widetilde{\kappa_{LS}}, \widetilde{CP.}(\mathbf{x}_{i,j})), \tag{29}$$

where we have used the symbol $\cdot$ replacing $\odot$ because any closest point algorithm can be used.

This method drastically reduces the error of curvature along the normal, depending on the κ_{LS} approximation and the interpolation errors, as shown in section 4.2.

Colinearity criterion improvement Recalling equation (13), for any point $\mathbf{x}$, its closest point $CP(\mathbf{x})$ on the surface Γ is collinear to the normal $\mathbf{n}_\Gamma(CP(\mathbf{x}))$ of that surface, or equivalently orthogonal to the tangent:

$$\overrightarrow{\mathbf{x}\,CP(\mathbf{x})} \cdot \boldsymbol{\tau}_\Gamma(CP(\mathbf{x})) = 0$$

Here we used the notations $\mathbf{n}_\Gamma$ and $\boldsymbol{\tau}_\Gamma$ to distinguish between the normal and tangent defined on the surface Γ and in the whole Eulerian domain. In a level set framework, we can compute the cosine angle between a vector $\overrightarrow{\mathbf{xy}}$ and the tangent at $\mathbf{y}$ with:

$$\omega(\mathbf{x}, \mathbf{y}) = \frac{\overrightarrow{\mathbf{xy}}}{|\overrightarrow{\mathbf{xy}}|} \cdot \boldsymbol{\tau}_\Gamma(\mathbf{y}).$$

Algorithm 1 does not guarantee the colinearity property. We propose Algorithm 2, noted $CP_\perp$, that converges to colinearity while still remaining on the surface. This algorithm can be understood as finding a candidate closest point to $\mathbf{x}$ on the interface and then correcting it toward the direction of colinearity. As this new corrected candidate may not be on the interface, we need to project it onto the interface. We apply this procedure iteratively until both the distance to the interface and the angle are smaller than a given threshold. A schematic view is proposed in Fig. 6.

Algorithm 2 $CP_\perp$: first order Newton method for the closest point search in a level set framework with colinearity property.

1. Initialize $i = 0$,
2. Find a first approximation of the closest point $\hat{\mathbf{y}}^0 = \widetilde{CP_{\odot}}(\mathbf{x})$
3. While $|\omega(\mathbf{x}, \hat{\mathbf{y}}^i)| > \epsilon$ do
   (a) $\hat{\mathbf{y}}^{i+1} = \widetilde{CP_{\odot}}(\hat{\mathbf{y}}^i + \omega(\mathbf{x}, \hat{\mathbf{y}}^i)\,\boldsymbol{\tau}(\hat{\mathbf{y}}^i))$
   (b) $i \leftarrow i + 1$
4. $\widetilde{CP}_\perp(\mathbf{x}) = \hat{\mathbf{y}}^i$

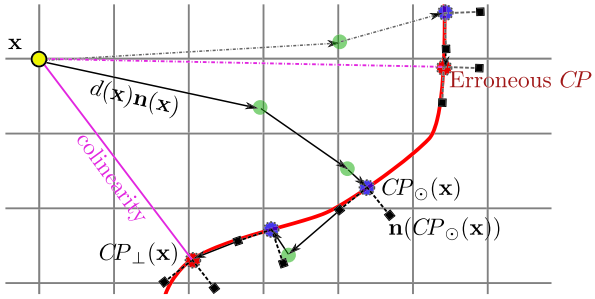


Fig. 6. Schematic view of the Closest Point algorithms. Errors on gradient vectors have been exaggerated and redundant notations omitted for the sake of clarity. Green dots stand for intermediate points, blue dots for $CP_{\odot}$ results and the red dot for the final $CP_{\perp}$ result. Normals on the surface are drawn as dotted square ended segments. The top gray dashed paths show an erroneous closest point due to poor approximations of the direction $\nabla\phi$ toward the surface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Remarks on the convergence of the algorithm and its computational cost This algorithm works under the assumption that ϕ is a smooth enough function where the initial closest point computed with $CP_{\odot}$ will not bring us close to a local minimum ensuring colinearity while not being the closest point on the surface, as shown by the dashed paths in Fig. 6. This will be the case particularly when ϕ is not close to a distance function ($\nabla\phi$ might pointing the wrong direction) and/or for points far from the surface. More efficient and precise numerical methods can be used to calculate the closest point. However, as presented in the numerical results, the proposed algorithm is adequate for computing accurate surface tension driven simulations.

In comparison to the previous algorithm, this one will compute more iterations. Ensuring the colinearity property greatly improves the accuracy of the Closest Point search and the curvature interpolation. We refer the reader to section 4.2 for numerical results and efficiency comparison.

Reinterpolating The extended field obtained by interpolation at the Closest Point can suffer from important variations of the interpolated field in the few cells around the surface. In order to reduce this perturbation and further homogenize the extended field, we perform a second interpolation process on the first field's extension:

$$\begin{aligned}\tilde{\kappa}_{CP^2_{\perp}}(\mathbf{x}_{i,j}) &= Interpolate(\widetilde{\kappa_{CP_{\perp}}}, \widetilde{CP}_{\perp}(\mathbf{x}_{i,j})),\\ \tilde{\kappa}_{CP^2_{\perp}}(\mathbf{x}_{i,j}) &= Interpolate(Interpolate(\widetilde{\kappa_{LS}}, \widetilde{CP}_{\perp}(\mathbf{x}_{i,j})), \widetilde{CP}_{\perp}(\mathbf{x}_{i,j})),\end{aligned} \tag{30}$$

which is the algorithm that we used in our numerical simulations.

This process only costs one more interpolation per cell as we have already computed the closest point $\widetilde{CP}_{\perp}(\mathbf{x}_{i,j})$.

We report in sections 4.2 and 4.3 the numerical results obtained with this method for geometrical and dynamic cases.

Remarks and discussion

- In the remainder of the article, if not stated explicitly, when noting CP, we assume the reinterpolation algorithm has been used, dropping the exponent 2 and the index $\perp$ for simplicity.
- The errors appearing in the numerical closest point computation compared to the exact closest point are due to:
  1. The Newton's descent first order discretization. It has to be noted that this first order descent is optimal in the case of a signed distance function where $|\nabla\phi| = 1$ over the whole domain.
  2. The interpolation functions used.
- In the case where ϕ is a signed distance function, the $CP_{\odot}$ and $CP_{\perp}$ will theoretically lead to exactly the same point on the surface as $\nabla\phi$ points to the exact closest point. However, due to the discretization of the Newton's descent and numerical and interpolation errors, the Closest Point computed will not be the same and $CP_{\perp}$ will guarantee colinearity.
- The calculation of the Closest Point in the whole domain – or a band around the interface – gives a straightforward result for reinitialization of the level set. The order of precision of the signed distance function reconstructed in this way depends on the precision of the CP algorithm used. To attain 4th order precision on the curvature one would need a precision of at least order 6 in the signed distance function. However, as reinitialization is not the topic of the current article, we will not detail the subject further.

3.4.5. Curvature computation on the staggered grid

On a staggered grid, the level set values and the curvature are computed at the center of the cells while the velocities and forces are computed at the center of the faces. Hence, in order to compute the surface tension term $\sigma\kappa\nabla c$, it is necessary to interpolate κ at these positions.

Given the Closest Point algorithm, one can easily interpolate the curvature field at any position of the computation domain, for example for the face between $\mathbf{x}_{i,j}$ and $\mathbf{x}_{i+1,j}$ noted $\mathbf{x}^f_{i+\frac{1}{2},j}$:

$$\kappa^f_{i+\frac{1}{2},j} = \widetilde{\kappa}_{CP.}(\mathbf{x}_{i+\frac{1}{2},j}) = Interpolate\left(\widetilde{\kappa_{LS}}, \widetilde{CP}.(x_{i+\frac{1}{2},j})\right)$$

or use the reinterpolation scheme as described above. However, this operation is at the cost of 2 (resp. 3) times more closest point computation in 2D (resp. 3D) which will affect the overall computational time. Alternatively, one can use interpolation to compute the curvature from the center of cells to the faces with adequate precision:

$$\kappa^{f}_{i+\frac{1}{2},j} = \frac{\kappa_{CP.\,i,j} + \kappa_{CP.\,i+1,j}}{2} \tag{31}$$

for a second-order accurate interpolation and

$$\kappa^{f}_{i+\frac{1}{2},j} = \frac{-\kappa_{CP.\,i-1,j} + 9\kappa_{CP.\,i,j} + 9\kappa_{CP.\,i+1,j} - \kappa_{CP.\,i+2,j}}{16} \tag{32}$$

for fourth-order accuracy. The later formula has a wider stencil of 4 points where the curvature is needed. It thus requires a larger band around the interface inside which the closest points has to be computed, consequently increasing the overall computational cost.

In our numerical simulations, we have found that using equation (31) is a good balance between accuracy and efficiency.

4. Numerical results

4.1. Foreword

In all our test cases, the order of convergence for the L_2 and L_∞ norms are the same, thus we will only show and discuss the latter norm which is the more restrictive. For this study, we used smooth representation of the surface Γ and the characteristic phase function via δ_ϵ and H_ϵ from equations (9) and (10) with $\epsilon = 2h$ where h is the spatial discretization step. This regularization contains approximately 81% of the Dirac mass in the 3 cells around the interface, i.e. for $\phi \in [-h, +h]$. Thus, all the error measures are numerically computed around the interface in a 1 cell neighborhood, fixing $N = 1$ in eq. (17). The thresholds used for the CP algorithms are set to h^4 for the distance and the angle criterion in order to attain fourth-order errors, as shown in the results below.

4.2. Geometrical convergence analysis of the method

In the following paragraphs, we study the Eulerian discretization errors for a fixed geometry by varying the methods used: the level set curvature estimation, the osculatory circle approximation and the Closest Point extension. The numerical convergence analysis for the CP methods are detailed for the ellipse in §4.2.2 which is the most discriminating.

4.2.1. Circle case

A circle of radius $R = 0.2$ is centered at the origin of a unit square. The level set is initialized as the exact signed distance function:

$$\phi(\mathbf{x}) = |\mathbf{x}| - R.$$

Here the relevant parameter is the ratio R/h representing the number of grid cells for a radius.

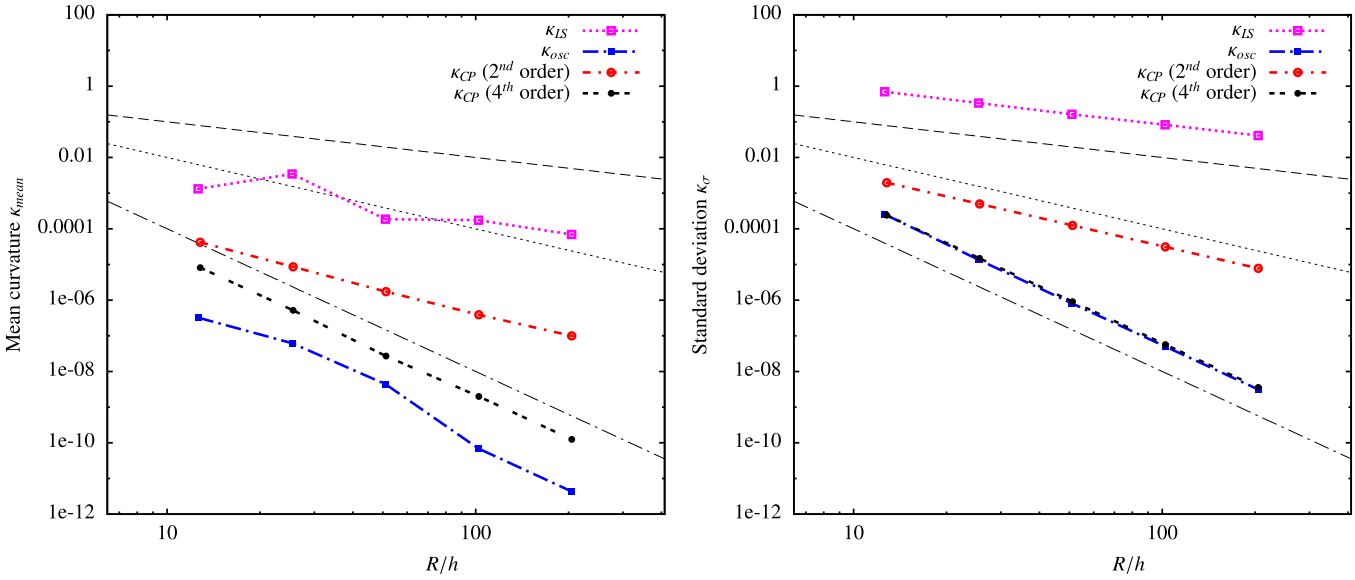
In this particular case the $CP_\odot$ and $CP_\perp$ methods give the same results so we will not distinguish them. This is due to the fact that for any point in the domain, by construction, the level set's gradient is always collinear to the normal at its closest point on the surface.

Order 2 and order 4 schemes. First, we show the importance of a high-order numerical computation of κ_{LS} from eq. (27). For that purpose, we have used classical central finite difference schemes of order 2 and 4 for gradients and Laplacian operators. Fig. 7 depicts the convergence of the different error measures relative to spatial discretization, taking $\kappa_{ex} = 1/R = 5$.

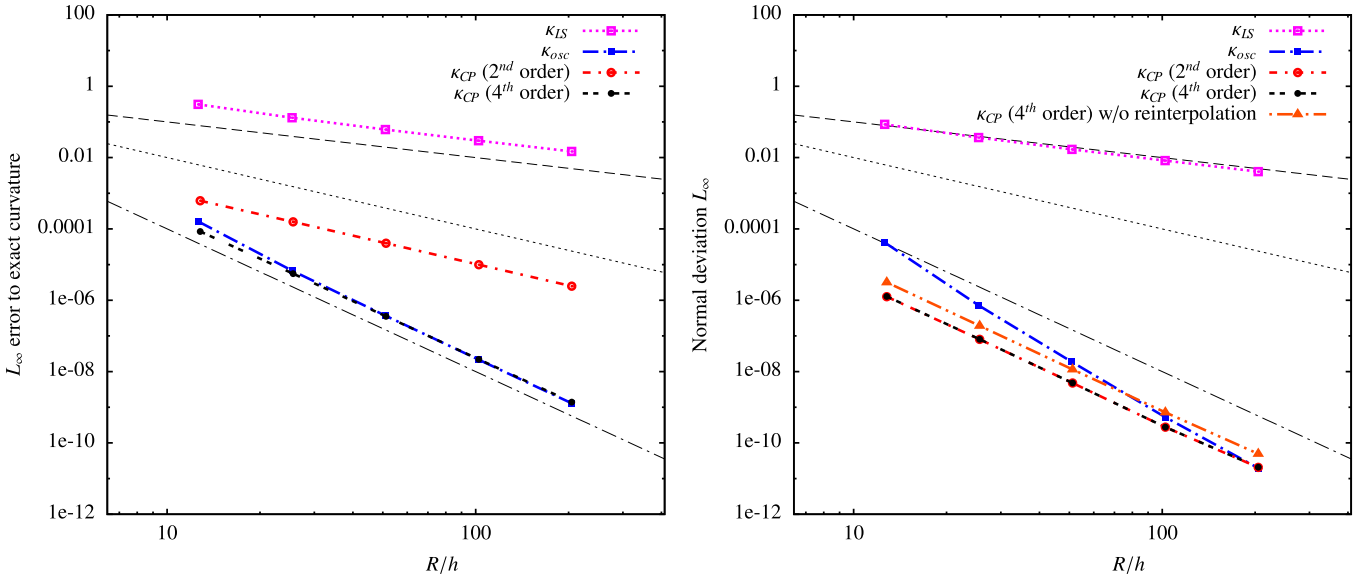
The results show a roughly first order convergence for the error measurements for κ_{LS}; in that case, the second and fourth-order schemes give qualitatively similar results. This is expected as the errors in the curvature come from the stretch/shrink effect as $\kappa_{osc} = \frac{1}{\kappa_{ex}^{-1} - d}$ away from the level set at $\phi = 0$, which dominates the error measures.

The CP method acts as expected, leading to the same order of convergence as the scheme used for the κ_{LS}. The normal deviation leads to fourth-order error convergence. This error is due to the interpolation functions used to compute the normal error deviation as discussed below. Concerning the CP reinterpolation method from eq. (30), the error measures are only different from the standard $CP_\perp$ method for the normal deviation: in that case we clearly see that reinterpolation of the curvature from a first curvature extension reduces the normal error by a factor of 3. As we will see in the ellipse test case below, this gain only holds for smooth enough curvatures without deteriorating the curvature field in the general case. However it reduces the spurious currents studied in §4.3.

The osculatory approximation based on a fourth-order scheme for κ_{LS} gives very good results: it converges at the same rate as the CP method for all the error measures except the normal deviation for which it converges more rapidly (around order 5). However, the absolute value of the normal deviation is approximately 100 times higher for $R/h \approx 12$. Despite this



(a) Mean curvature error and standard deviation.



(b) L_∞ error to exact curvature and L_∞ normal deviation.

Fig. 7. Circle geometrical test case: spatial convergence as a function of R/h for the 4th order κ_{LS}, the osculatory circle correction and the CP method with order 2 and order 4 κ_{LS}. κ_{LS} errors are only shown for $O(h^4)$ as the results with the $O(h^2)$ scheme are almost identical.

Table 2
Circle geometrical test case: approximate order of convergence of error measures for the different methods.

Method	L_∞	$\frac{\lvert\kappa_{mean}-\kappa_{ex}\rvert}{\kappa_{ex}}$	Std. dev. $\tilde{\kappa}_\sigma$	Normal $L_{\infty,\tilde{\kappa}_{\sigma,\mathbf{n}}}$
κ_{LS} 4th order	1	1	1	1
κ_{osc} (w/ κ_{LS} 4th order)	4	4	4	5
κ_{CP} (w/ κ_{LS} 2nd order)	2	2	2	4
κ_{CP} (w/ κ_{LS} 4th order)	4	4	4	4

good convergence rate, it is important to recall that the useful properties of this method rely on the first order approximation of the local curvature to the osculatory circle which will be deteriorated in the general case (i.e. contrary to this ideal circle case) as it will be shown in the next sections for the ellipse. The convergence rates are shown in Fig. 7 and summarized in Table 2.

The aliasing that can be observed in the mean curvature (Fig. 7(a)) is due to the fact that we use a sharp stencil $\overline{\delta_\Gamma}$ for the error measurements around the interface, affecting any measurement contrary to a smooth δ_Γ.

Even though using order 4 schemes for κ_{LS} yields much better results, it is important to note that it has to be avoided for fast varying ϕ, i.e. when the local curvature is of the order of h, where it would raise higher errors than a more compact

scheme because of high-frequency amplified errors. We chose a sufficiently large R/h ratio to avoid such perturbations. However, in the general case, one could think of mixing different schemes based on a first estimation of the curvature.

Errors in the curvature estimation in the presence of fictitious perturbations. In order to see how the method responds to numerical errors that will arise with the transport of the level set, we introduced small perturbations to the initial level set such as:

$$\hat{\phi}(\mathbf{x}) = \phi(\mathbf{x}) + e(h^M) \tag{33}$$

where $\phi(\mathbf{x})$ is the signed distance function to the circle and $e(h^M)$ is a random function that takes its values in the interval $[-h^M, +h^M]$. One has to note that if $e(h^M) = h^M$ is constant then $\hat{\phi}(\mathbf{x}) = \phi(\mathbf{x}) + h^M$ and the relative error for the level set function is

$$\frac{\hat{\phi}(\mathbf{x}) - \phi(\mathbf{x})}{\phi(\mathbf{x})} = \frac{h^M}{\phi(\mathbf{x})} = O(h^{M-1})$$

near the interface. Thus if we add a perturbation of order 3 then the expected perturbation on the curvature computation should be of the order $3 - 1 - 2 = 0$. However, in the following results, as $e(h^M) \in [-h^M, +h^M]$ is a random function, we expect a smaller deterioration on the order of convergence.

Here we used a fourth-order scheme for κ_{LS}. Fig. 8 shows that the introduction of a $O(h^3)$ (resp. $O(h^4)$) perturbation approximately reduces the order of convergence from 4 to 1 (resp. from 4 to 2), as expected.

4.2.2. Ellipse case

In order to test the robustness of the method in a more general case, we applied the same spatial convergence study to an ellipse. Particularly, we will show the effect of the collinear closest point method compared to the standard method. The surface is implicitly defined by the level set:

$$\phi(\mathbf{x}) = \sqrt{\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2} - R$$

with $a = 1.2$, $b = 0.8$ and $R = 0.2$, or in a parametric form:

$$\Gamma(\theta) = (aR\cos(\theta), bR\sin(\theta)).$$

It should be noted that the level set function ϕ is not a signed distance function which is not trivial to compute. The exact curvature extension that was constant for the circle case is here less trivial. Recall that the curvature extrapolation is:

$$\kappa_{ex}(\mathbf{x}) = \kappa_{ex}(CP_{ex}(\mathbf{x})) = \kappa_{\Gamma,ex}\left(\theta_{CP_{ex}}(\mathbf{x})\right),$$

where $\theta_{CP}(\mathbf{x})$ is the angular parameter for the closest point to $\mathbf{x}$ on the surface. We need to compute the exact closest point $CP_{ex}(\mathbf{x})$ on the ellipse which is obtained by solving the equation

$$\overrightarrow{\mathbf{x}\,CP_{ex}(\mathbf{x})} \cdot \Gamma(\theta) = 0$$

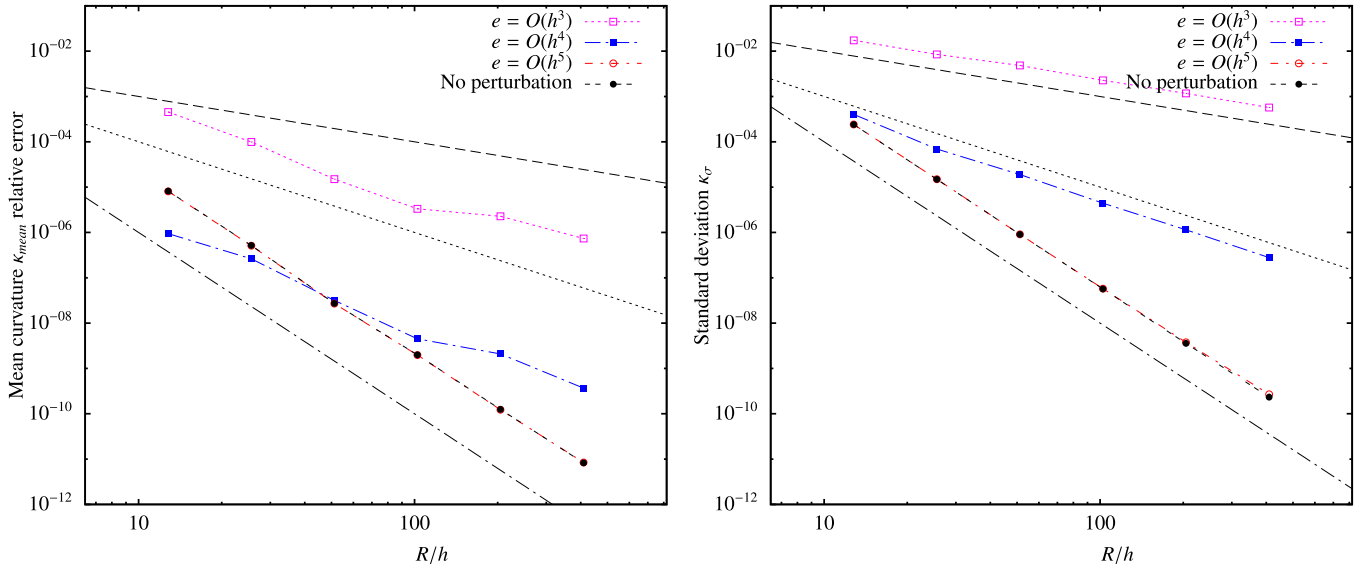
for θ, as the line from $\mathbf{x}$ to its closest point on Γ is collinear to the normal at that point. This non-linear equation has in the general case 2 solutions but only one in the nearest quadrant (known by the position of $\mathbf{x}$ compared to the ellipse's parameters). To solve it we used Newton's method with the search reduced to the nearest quadrant. The curvature at that point $\Gamma(\theta)$ is then computed with:

$$\kappa_{\Gamma,ex}(\theta) = \frac{ab}{\left(b^2\cos^2(\theta) + a^2\sin^2(\theta)\right)^{3/2}}.$$

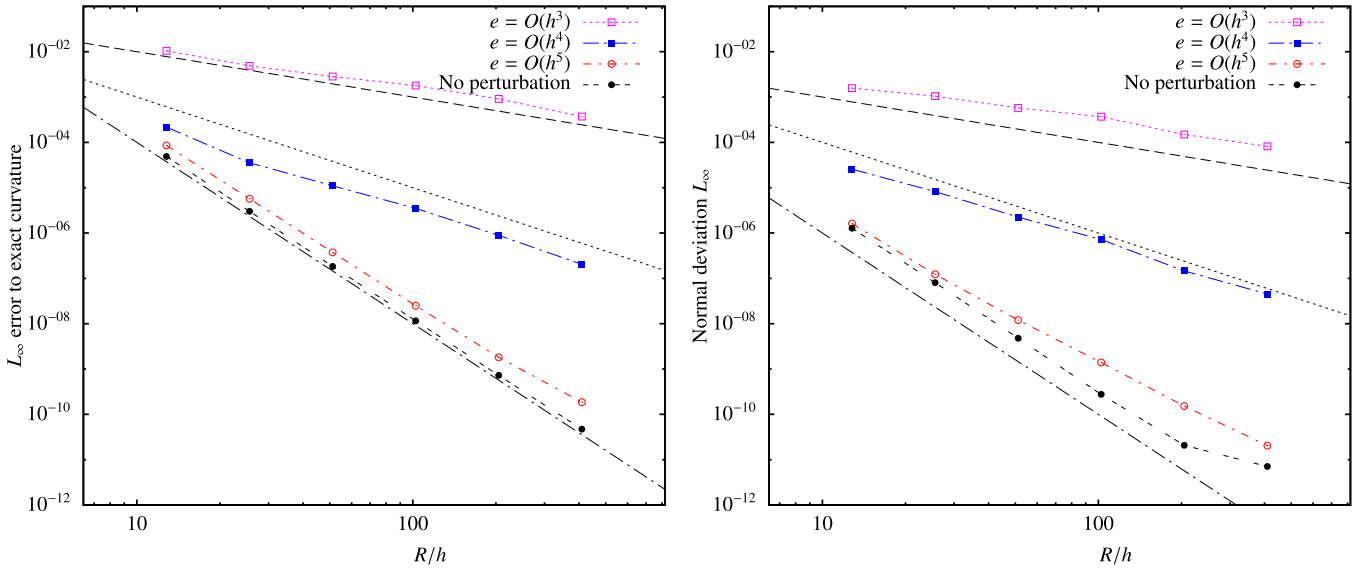
Note that this "exact" extended curvature is subject to discretization errors due to Newton's descent. In consequence the errors that we present below can be biased by this additional perturbation. However we expect this to be negligible.

Also, as stated previously in §2.4.2 comparing the mean curvature and the standard deviation is meaningless since the curvature varies along the surface, hence we removed those two error measures for this ellipse test case.

Closest point spatial convergence. In this paragraph, we study the spatial convergence of the two different CP algorithms. For this purpose, we look at the error in the distance map defined as the distance from $\mathbf{x}$ to its computed closest point on the surface $CP(\mathbf{x})$, and the distance between $CP(\mathbf{x})$ and the exact closest point $CP_{ex}(\mathbf{x})$. Those two errors have different meanings. While the error in the distance to the surface is meaningful, as we aim to interpolate values on the interface, the error in the closest point search is, in our case, more relevant.



(a) Mean curvature relative error and standard deviation.



(b) L_∞ error to exact curvature and L_∞ normal deviation. The last normal deviation measure for the calculus without perturbation is decreasing less rapidly because of real number double precision limits at such small scales ($h = 1/2048$ and errors of the order 10^{-11}).

Fig. 8. Circle geometrical test case: convergence of the different error measures as a function of R/h in presence of small perturbations for the fourth-order *CP* method.

The L_2 and L_∞ closest point error measures are defined as

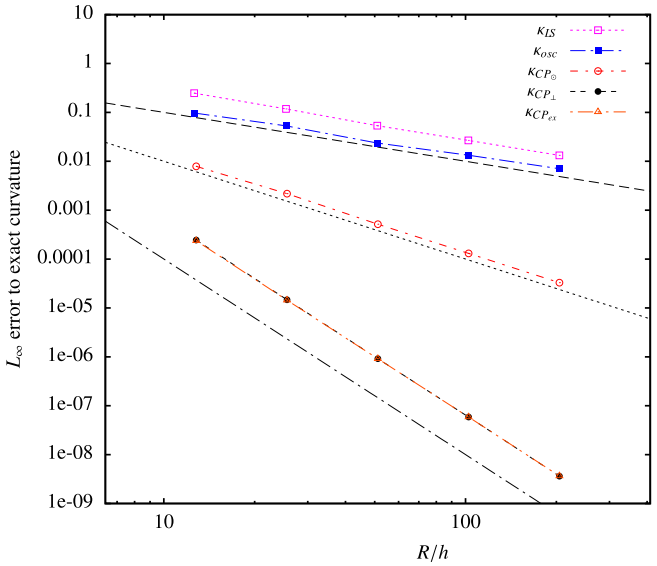
$$L_{2,CP.} = \sqrt{\int |CP.(\mathbf{x}) - CP_{ex}(\mathbf{x})|^2}$$

and

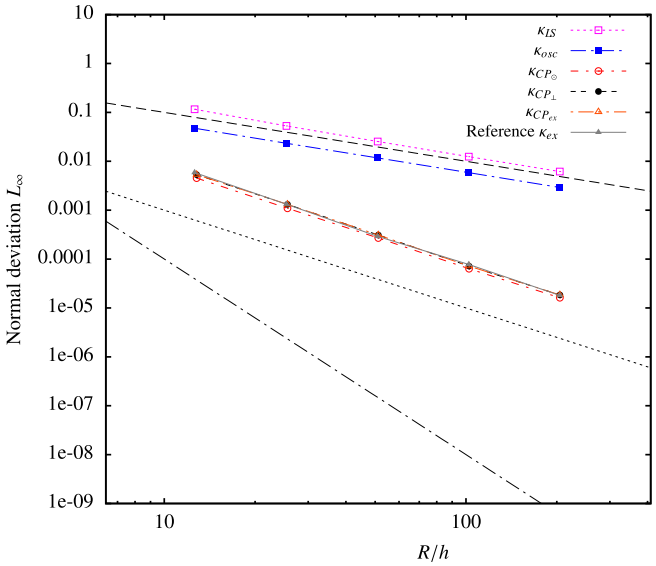
$$L_{\infty,CP.} = \max |CP.(\mathbf{x}) - CP_{ex}(\mathbf{x})|$$

where the dot index is either $\odot$ or $\perp$ standing for the two different CP algorithms. We also define the distance map error measures as

$$L_{2,d.} = \sqrt{\int (d(\mathbf{x}, CP.(\mathbf{x})) - d(\mathbf{x}, CP_{ex}(\mathbf{x})))^2}$$



(a) L_∞ errors to exact curvature. The last three curves stand for the three closest point estimates: $CP_\odot$, $CP_\perp$ and CP_{ex}. The use of $CP_\perp$ lead to a gain of 2 orders in the convergence and gives results similar to those of CP_{ex} for reference.



(b) Normal deviation with the lower bound reference curve using κ_{ex} which equals all the CP methods.

Fig. 9. Ellipse geometrical test case: spatial convergence as a function of R/h for κ_{LS} with an order 4 scheme, the osculatory circle correction and the $CP_\odot$ and $CP_\perp$ methods.

Table 3
Ellipse geometrical test case: approximate order of convergence of error measures for the different methods, a 4th order κ_{LS} computation is used.

Method	L_∞	Normal $L_{\infty,\tilde{\kappa}_{\sigma,\mathbf{n}}}$
κ_{LS}	1	1
κ_{osc}	1	1
$CP_\odot$	2	2
$CP_\perp$	4	2
CP_{ex}	4	2

and

$$L_{\infty,d_\cdot} = \max |d(\mathbf{x}, CP_\cdot(\mathbf{x})) - d(\mathbf{x}, CP_{ex}(\mathbf{x}))|$$

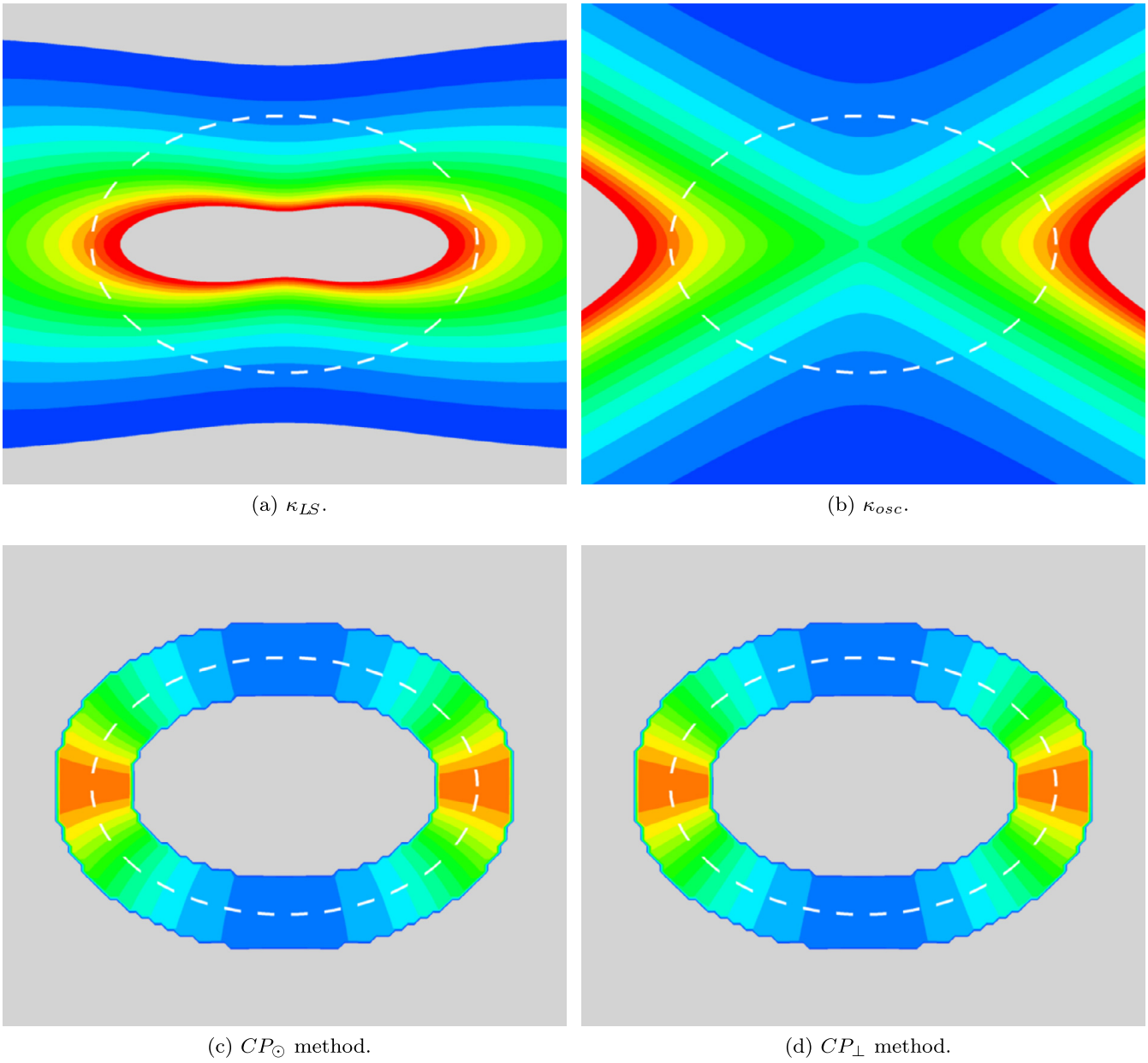
where $d(\mathbf{x}, \mathbf{y})$ is here the Euclidean distance from $\mathbf{x}$ to $\mathbf{y}$. In a similar fashion to §3.4.2 error measures are numerically evaluated in a band around the surface.

Curvature spatial convergence. Fig. 9 shows the spatial convergence for the computed curvature as a function of R/h for the L_∞ error measures (L_2 errors are qualitatively equivalent). Table 3 presents an overview. As expected, the osculatory circle method is decreased to first order while the CP method without colinearity converges at second-order, and the use of $CP_\perp$ converges at fourth-order showing the clear advantage of using an accurate Closest Point search. Moreover, this last method approximately equals the reference results when using CP_{ex}. In the ellipse case, the error measures obtained with and without reinterpolation of the level set (eq. (30)) are approximately equal and hence not shown. We can conclude that this process does not deteriorate the solution in the general case, while it has a clear benefit for smooth surfaces locally homogeneous to a circle.

In this test case the L_∞ normal errors are decreased to second-order for both CP methods. This loss of precision compared to the circle case is due to the variation of curvature created by the interpolations from equation (25). This can be observed in Fig. 9(b) where we have also plotted the normal deviation error based on the interpolation of the exact curvature computation $\kappa_{ex}(\mathbf{x})$ at $CP_{ex}(\mathbf{x})$, which is the numerical lower bound that can be attained by any method. All the results using any of the CP methods approximately equal this reference curve for $L_{\infty,\tilde{\kappa}_{\sigma,\mathbf{n}}}$.

Fig. 10 shows a spatial representation of the curvature in the domain for the different methods for $R/h = 25.4$. We can clearly see the effect of the CP algorithm on the extension of the curvature along the normal. Looking closely, the simple $CP_\odot$ method induces non-rectilinear iso-values for the curvature while the $CP_\perp$ method corrects this problem. The exact curvature profile obtained with CP_{ex} is not shown as it is indistinguishable from the $CP_\perp$ method.

Numerical convergence in the presence of perturbations. Fig. 11 shows the spatial convergence of the closest point error and the distance map as a function of R/h with varying perturbations applied to the level set as presented in eq. (33). Summarized



(a) κ_{LS}. (b) κ_{osc}.

(c) $CP_{\odot}$ method. (d) $CP_{\perp}$ method.

Fig. 10. Ellipse geometrical test case: curvature profiles for the different methods. The position of the ellipse is drawn with white dashed lines. The color variation goes from blue ($\kappa = 2.5$) to red ($\kappa = 10$), gray zones represent curvature values outside that interval or outside the computation band for the *CP* methods. The $CP_{\perp}$ method and the exact curvature profiles are indistinguishable hence the latter is not shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in Table 4, we observe a maximum order 2 for the closest point error measure in the case of the $CP_{\odot}$ algorithm while it increases to fourth-order for the $CP_{\perp}$ method. The $CP_{\odot}$ method's distance map has a maximum third-order convergence, while the $CP_{\perp}$ one is up to fourth-order. We believe that the convergence rate is limited because of the errors on the computation of $\nabla\phi$, on the numerical thresholds used, but more importantly because of the fourth-order interpolation functions used. However this fourth-order convergence rate is sufficient in our case and for the applications targeted.

The $CP_{\perp}$ algorithm we propose in this paper shows a clear benefit compared to the $CP_{\odot}$ used in the literature that will have an important impact on the order of convergence of the κ_{CP} methods studied in the next paragraph.

Computational cost of the closest point algorithms. Fig. 12 shows the mean number of iterations per grid point for the $CP_{\odot}$ and $CP_{\perp}$ methods. The use of a fourth-order accurate algorithm requires less than 3 times more iterations than using a second-order accurate. This difference is mostly due to the reduced threshold ϵ from h^2 to h^4 that is more hardly reached. With fourth-order precision, for coarse grids with more errors on the computation of the normal and interpolations, the $CP_{\perp}$ algorithm requires significantly more iterations than the simple $CP_{\odot}$ algorithm (around three times for $R/h = 12.8$). This ratio reduces to around 1.6 for a finer mesh. This behavior is expected as the higher the curvature to spatial step ratio, the farther away the first $CP_{\odot}$ iteration will bring us from the exact closest point. Fig. 13 reports the computation costs in

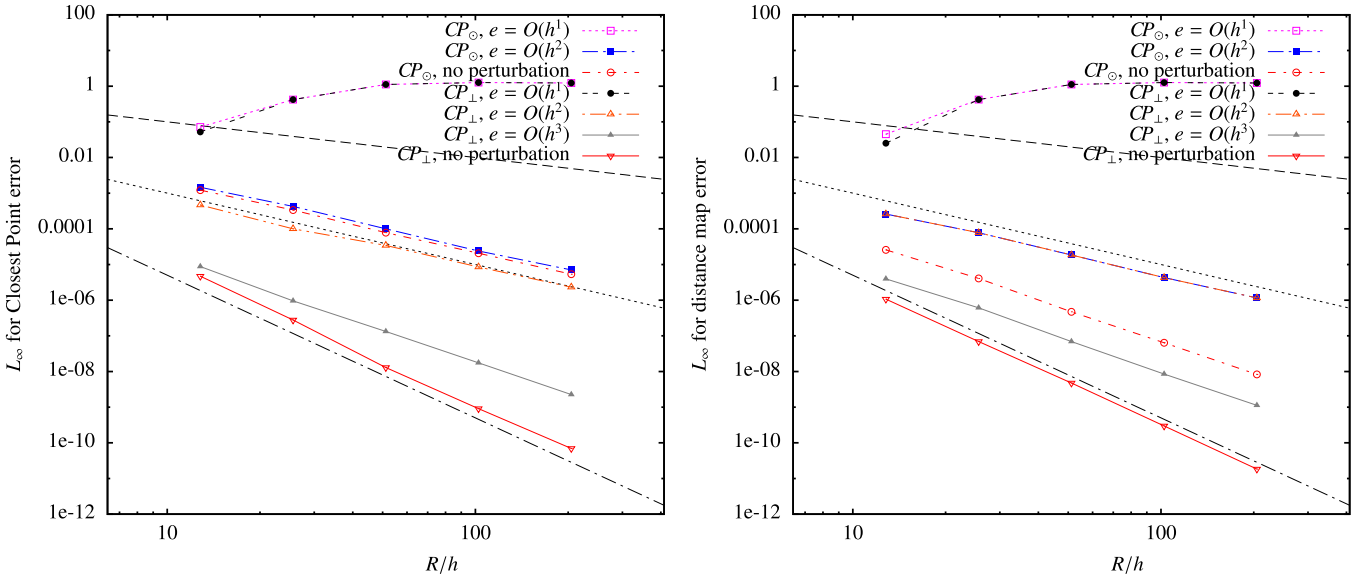
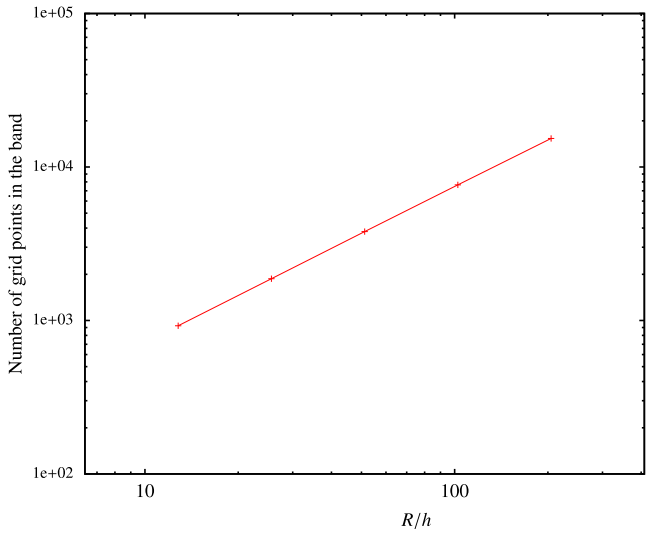


Fig. 11. Ellipse geometrical test case: convergence of the $L_{\infty,CP}$ closest point error (left) and $L_{\infty,d}$ distance map error (right) as a function of R/h in the presence of small perturbations for $CP_\odot$ and the $CP_\perp$ methods. Results for $CP_\odot$ with $O(h^3)$ and $O(h^4)$ errors are almost equal to the ones with no perturbation thus not shown. Similarly, results for $CP_\perp$ with $O(h^4)$ perturbations are not shown.

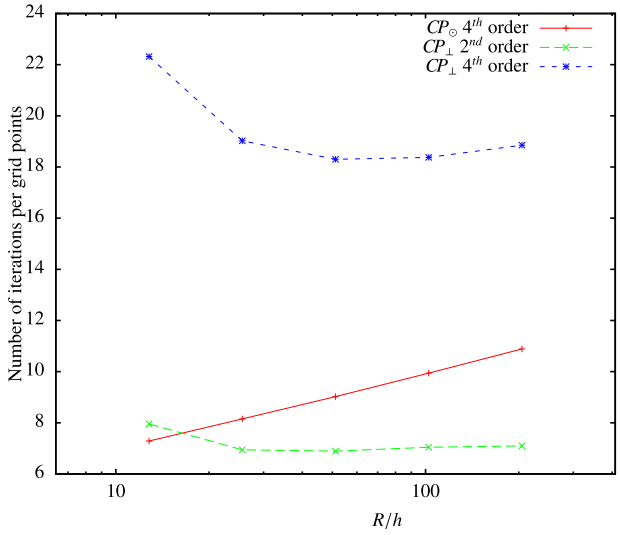
Table 4
Ellipse geometrical case: approximate orders of convergence of error measures for the two different CP methods and ϕ perturbations.

Method	Perturb.	$L_{\infty,CP}$	$L_{\infty,d}$
$CP_\odot$	$O(h^1)$	0	0
$CP_\odot$	$O(h^2)$	2	2
$CP_\odot$	$O(h^3)$	2	3
$CP_\odot$	$O(h^4)$	2	3

Method	Perturb.	$L_{\infty,CP}$	$L_{\infty,d}$
$CP_\perp$	$O(h^1)$	0	0
$CP_\perp$	$O(h^2)$	2	2
$CP_\perp$	$O(h^3)$	3	3
$CP_\perp$	$O(h^4)$	4	4



(a) Linear increase of the number of points in the $6h$ band with respect to R/h.

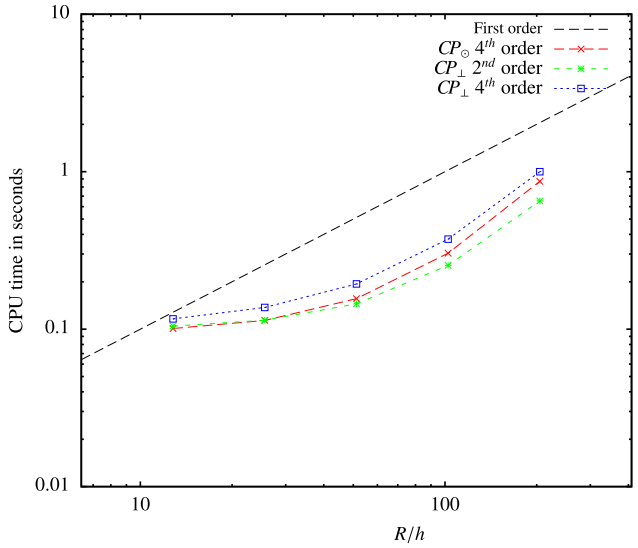


(b) Mean number of iterations per grid points in the $6h$ band for the closest point algorithms $CP_\odot$ with fourth-order precision and $CP_\perp$ with second and fourth-order precision.

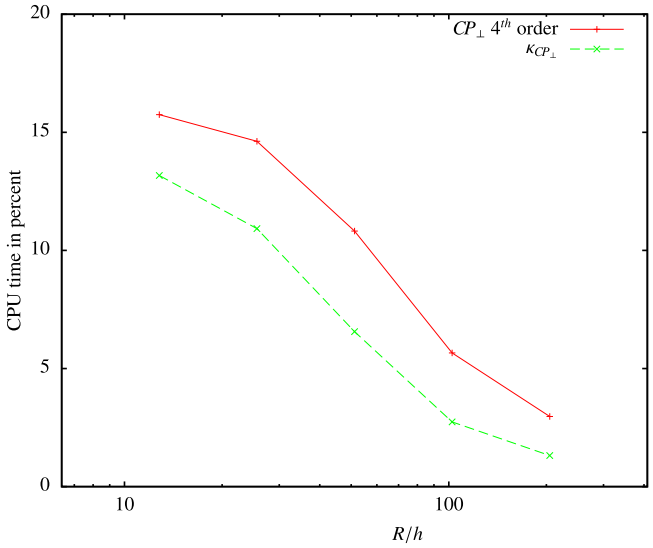
Fig. 12. Efficiency of the closest point algorithms.

terms of CPU time for the different methods. The use of the fourth-order accurate $CP_\perp$ algorithm against the second-order one (the quicker) requires only 1.5 times more computational time for the finer mesh. Moreover, in our fluid solver, the most accurate algorithm represents less than 3% of the overall computational time for one iteration.

As the number of grid points grows linearly with the spatial discretization, the use of the $CP_\perp$ with fourth-order precision is highly recommended and has a low impact on the computational time.



(a) Fourth-order $CP_\odot$, $CP_\perp$ with second and fourth-order precision.



(b) Percentage of the computational time for one fluid solver iteration for the closest point computation and the curvature computation.

Fig. 13. Computational cost of the CP algorithms as a function of the spatial discretization.

4.2.3. Conclusion of the geometrical study

Through this geometrical study on the circle and ellipse cases, we have shown that the CP method attains high order (up to 4) spatial convergence on the different error measures guaranteeing numerical accuracy for Criteria 2.1, 2.2 and 2.3 of §2.4 and thus for the surface tension driven simulations. The use of a fourth-order κ_{LS} computation is also a key in attaining this precision and is very important in the reduction of errors in dynamic numerical simulations.

Compared to the $CP_\odot$ method, the $CP_\perp$ method gains 2 orders of convergence for the curvature's L_2 and L_∞ error in the general (ellipse) case while only requiring double the number of interpolations. The reinterpolation $CP^2_\perp$ method improves the error in the normal deviation for the circle, which will help stabilize the static column equilibrium case studied below.

4.3. Static viscous column equilibrium with constant density

The problem of the equilibrium of a 2D column subjected to surface tension forces has been widely used as a reference in the literature, for example [39,16,40] propose numerical results and analysis with various methods in VOF or level set frameworks. From the Laplace law, without discretization errors, the surface tension forces should not create any current – the fluid should be at rest – as they are at equilibrium with the pressure jump between the two phases:

$$\nabla p = \frac{1}{We}\kappa \mathbf{n}\delta_\Gamma \tag{34}$$

giving a pressure inside the drop of $p_{drop} = \sigma\kappa$ when the reference pressure outside is 0.

The Weber number is given as

$$We = Re \cdot Ca = \frac{\rho U L}{\mu} \cdot \frac{\mu U}{\sigma} = \frac{\rho U^2 L}{\sigma}, \tag{35}$$

with U (resp. L) a reference velocity (resp. length) characterizing the problem. In our problem, $L = D$, the diameter of the column.

Two other relevant non-dimensional numbers are the Laplace and the Ohnesorge number relating the surface tension to the viscous forces:

$$La = \frac{\sigma\rho L}{\mu^2} = Oh^{-2} = \frac{Re^2}{We}. \tag{36}$$

The Laplace number characterizes the numerical behavior of the flow computed due to numerical errors on the discretization of the surface tension forces, i.e. the errors in the localization δ_Γ of the column's interface, its normal $\mathbf{n}$ and its curvature κ will create capillary waves and thus currents that will propagate in the fluids through the viscous term. The more viscous the fluid is then the more smoothly the problem will reach equilibrium. At high Laplace numbers, we expect the interface between the two phases to oscillate/vibrate around its equilibrium state more rapidly and for a longer time.

In the following sections, for clarity, we call "velocity" the root mean square of the fluid velocity as:

$$v_{rms} = \sqrt{\int_{\Omega} |\mathbf{v}(\mathbf{x})|^2 d\mathbf{x}} \tag{37}$$

and the capillary number is defined in absolute value as:

$$Ca = \frac{\mu |\mathbf{v}|_{max}}{\sigma}. \tag{38}$$

If not stated differently, v_{rms} is given in the T_σ and U_σ reference frames as defined below.

As shown in [11], applying a constant curvature κ_{exact} for the surface tension force in a balanced force CSF framework leads to numerical equilibrium instantly, i.e. v_{rms} is of the order of the machine precision. We study below the effect of errors on the computation of the curvature.

Origins of the parasitic currents. In [17], the authors analytically demonstrate that, for the static column at equilibrium, the parasitic currents are quadratically proportional to the curvature. This is particularly crucial as the numerical errors will increase rapidly at small scales and finally dominate the flow. We push their study further by introducing the errors in the computation of the volume fraction gradient $\nabla c = \overline{\nabla c} + \beta$, where $\overline{\nabla c}$ is the exact solution and β represents the numerical errors that is not necessarily the gradient of a function. The curvature κ is decomposed in a similar way: $\kappa = \overline{\kappa} + \kappa'$, where κ' is the error part. Neglecting the viscous stress tensor and posing $\rho = 1$ for clarity, the Navier–Stokes equation holds:

$$u_t + u \cdot \nabla u = -\nabla p + \sigma \kappa \nabla c. \tag{39}$$

When the disk is at equilibrium, the pressure gradient and the surface tension force are in balance, i.e. $\nabla p = \sigma \overline{\kappa} \overline{\nabla c}$, equation (39) becomes:

$$u_t + u \cdot \nabla u = \sigma (\overline{\kappa} \beta + \kappa' \overline{\nabla c} + \kappa' \beta).$$

The error term $\kappa' \beta$ is of higher-order and, numerically, the error in the gradient of c is of higher-order than the error on κ because they are usually derived from the same surface representation. Hence, if we drop those two terms, we find back the solution proposed in [17]. This demonstrates the origins of parasitic currents for the disk at equilibrium, dominated by the errors in the curvature computation.

Non-dimensional reference frames. All the velocities are given in the velocity scale for the inviscid problem:

$$U_\sigma = \sqrt{\frac{\sigma}{\rho D}}$$

as well as the corresponding time scale:

$$T_\sigma = \sqrt{\frac{\rho D^3}{\sigma}}$$

which is proportional to the period of the capillary waves. We also define the viscous time scale as:

$$T_\mu = \frac{D^2}{\mu}.$$

Time step restriction. As was clearly demonstrated in [41], the stability of flow computations subjected to surface tension is limited not by its explicit treatment but by constraints due to the physical propagation of waves. It yields to the following condition on the time step:

$$\Delta t_\sigma^{stat} \leq \frac{h}{\hat{c}_\sigma} \sqrt{\frac{(\rho_1 + \rho_2) h^3}{2\pi\sigma}}$$

for the static case and

$$\Delta t_\sigma^{dyn} \leq \frac{h}{\hat{c}_\sigma + u_\Sigma^k}$$

for the dynamic case, where u_Σ^k is the velocity of the flow at the interface parallel to the capillary wave of number k.

In most of the simulations studied below, we have respected the static time step constraint, however we have observed, like detailed in [41], that for problems that are sufficiently smooth, larger time steps can be used without introducing instabilities in the computations.

Table 5
Static column case: numerical results for the finest grid ($R/h = 102.4$) and approximate global order of convergence for the $CP^2_\perp$ method, order 2 and order 4 κ_{LS} calculation, $La = 120$. Detailed numerical results are given for varying R/h in Table 11.

(a) Velocity error measures and pressure error.

κ_{LS}	R/h	Ca at $t = \Delta t$	O	Ca at $t_\sigma = 30$	O	max Ca	O	$E(\Delta p_{mean})$ at $t_\sigma = 30$	O
Order 2 scheme	102.4	1.19×10^{-8}	1.98	1.34×10^{-14}	4.93	2.27×10^{-7}	1.99	9.91×10^{-7}	2.00
Order 4 scheme	102.4	3.74×10^{-11}	4.05	2.08×10^{-14}	4.30	4.13×10^{-10}	3.97	4.24×10^{-10}	3.94

(b) Curvature error measures at $t_\sigma = 30$.

κ_{LS}	R/h	$\lvert\frac{\kappa_{mean}-\kappa_{ex}}{\kappa_{ex}}\rvert$	O	L_∞ curvature error	O	κ_σ	O	L_∞ normal dev.	O
Order 2 scheme	102.4	9.91×10^{-7}	2.00	9.91×10^{-7}	2.00	6.31×10^{-11}	5.28	1.68×10^{-11}	5.09
Order 4 scheme	102.4	4.24×10^{-10}	3.94	4.47×10^{-10}	3.95	2.30×10^{-11}	4.93	1.27×10^{-11}	4.57

4.3.1. General configuration

We consider the physical problem of a column of diameter $D = 2R = 0.4$ centered at the origin of a unit square. The density and the viscosity are set to unity: $\rho = \mu = 1$. The Laplace number is thus varied by changing the surface tension coefficient σ.

The level set is initialized exactly as

$$\phi(\mathbf{x}) = |\mathbf{x}| - R.$$

No symmetry property is used; we compute solutions in the whole domain. No-slip conditions are applied to all boundaries.

The pressure error, considering the reference pressure outside the drop to be zero, is computed as:

$$E(\Delta p_{mean}) = \frac{|p_{2,mean} - p_{1,mean} - p_{drop}|}{p_{exact}}$$

where $p_{i,mean}$ is the mean pressure calculated in the ith fluid by integrating numerically in the whole domain, with $i = 2$ inside the column:

$$p_{i,mean} = \frac{\sum p \cdot H_{\epsilon,i}(\phi/|\nabla\phi|)}{\sum H_{\epsilon,i}(\phi/|\nabla\phi|)}$$

where $H_{\epsilon,i}$ is the regularized characteristic function associated with the ith phase.

4.3.2. Order 2 and order 4 κ_{LS} calculation

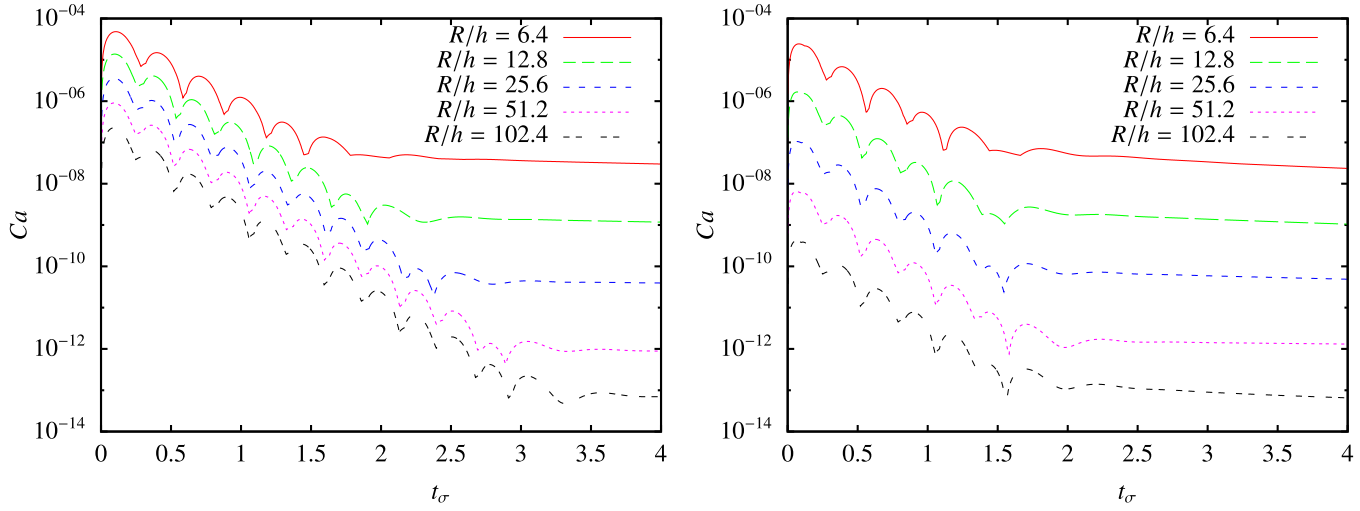
We study the dynamic effect of spatial errors due to the κ_{LS} computation precision on the $CP_\perp$ method by varying the scheme used and the number of grid points. The Laplace number is 120, corresponding to $\sigma = 300$, and the time step is set to $\Delta t = 3 \times 10^{-5}$ s. The curvature errors and the impact on the fluid velocity are shown in Fig. 14; as time evolution for the pressure, the mean curvature and its normal deviation are small, they are not shown. The numerical results are reported in details in Appendix A.2, Table 11, and summarized for the finest grid resolution in Table 5.

The results clearly show that, when compared to the second-order scheme, the fourth-order computation of κ_{LS} greatly reduces the capillary number in the first time steps to order 4 as well as decreases the maximum spurious currents. When the approximate numerical equilibrium is reached for all simulations, around $t_\sigma = 30$, the pressure error also converges at fourth-order, as expected from eq. (34) and from the balanced force CSF principle. After $t_\sigma = 30$, the velocity still declines but much less rapidly.

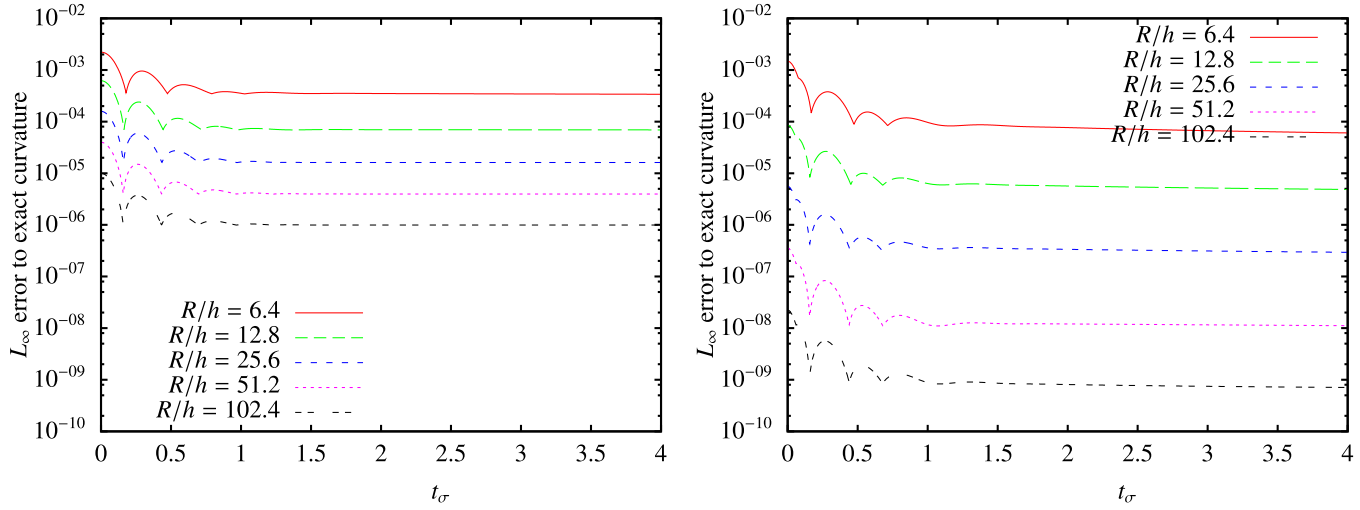
The capillary number converges toward machine precision at an even higher order when spatial discretization tends to zero. The RMS velocity follows qualitatively the same behavior thus we do not show it here. It is noticeable that the second and fourth-order schemes converge toward the same approximate equilibrium Ca and RMS velocity residual. This is due to the physical dynamics of the test case that searches for a numerical equilibrium state, which geometrically depends on h, through the level set function displacement by minimizing the standard deviation on the curvature. We see in Fig. 14(c) that for the same spatial step h, the approximate minimal κ_σ is qualitatively independent of the κ_{LS} accuracy. However, this minimum is reached at around $t_\sigma = 1.5$ for any discretization of the 4th order scheme, whereas that time is stretching with decreasing h for the second-order scheme. As with the latter scheme, more numerical errors are introduced in the early stages, and the numerical equilibrium is reached later implying more oscillations of the interface.

Surprisingly, we observe a faster convergence rate on the capillary number at $t_\sigma = 30$ for the second-order κ_{LS} computation. We believe that at very low resolution such small velocities are close to machine precision and thus very sensitive to numerical perturbations, which are non-negligible at these scales. For instance, the maximum velocities for the $R/h = 102.4$ discretization approximately equals 5×10^{-12} which is of the same order of magnitude as the maximum velocity introduced in the flow when applying the constant curvature $\kappa_{ex} = 1/R$.

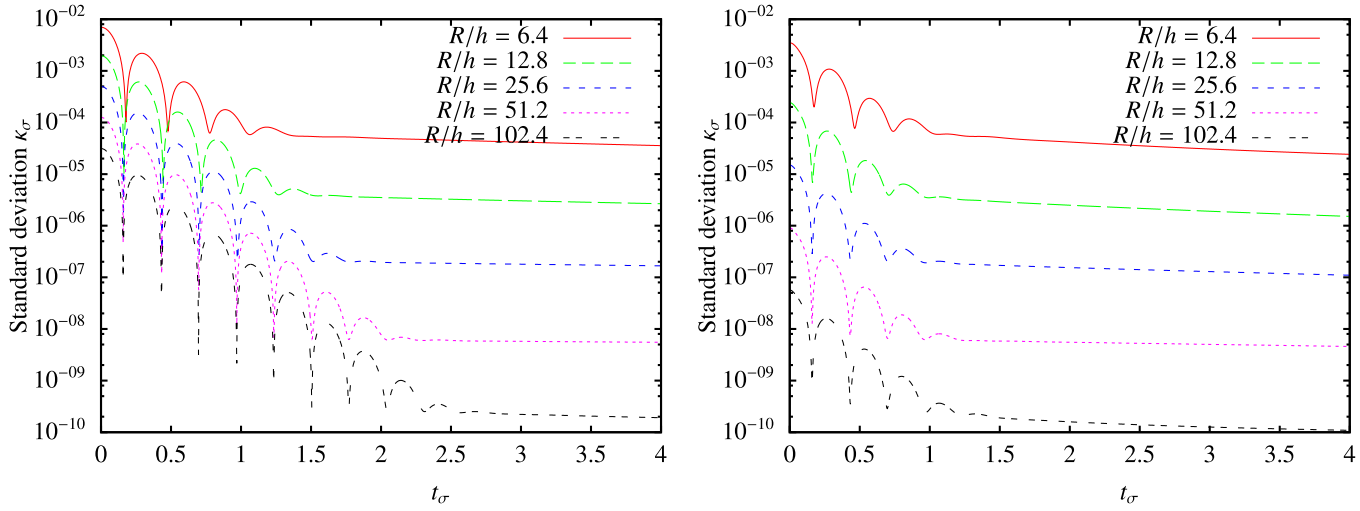
However, as the curvature and the surface are in better geometrical balance for high-order schemes for κ_{LS} (refer to 4.2 for the geometrical convergence analysis), the errors induced in the early time steps and on the maximum capillary number



(a) Capillary number.



(b) L_∞ error to exact curvature.



(c) Standard deviation to the mean curvature.

Fig. 14. Static column case: convergence study for the RMS velocity, L_∞ curvature error and standard deviation using the $CP^2_\perp$ method and order 2 (left) and order 4 (right) κ_{LS} calculation, $La = 120$.

Table 6
Static column case: convergence of capillary number in function of the Laplace number and R/h with the proposed method compared to results from [12,13,15,42].

(a) Varying Laplace number at $R/h = 6.4$.

R/h	Ca		La			
			120	1200	12 000	120 000
6.4	At $t_{cap} = 250$	Proposed method	6.74×10^{-9}	6.82×10^{-9}	5.42×10^{-7}	2.68×10^{-7}
		[42]	4.59×10^{-7}	7.80×10^{-7}	2.18×10^{-6}	3.00×10^{-6}
		[12] Cartesian	1.10×10^{-7}	1.20×10^{-7}	1.44×10^{-6}	3.09×10^{-6}
		[15]	2.18×10^{-6}	2.18×10^{-6}	2.22×10^{-6}	–
		[13]	5.71×10^{-6}	5.99×10^{-6}	8.76×10^{-6}	–
	At $t_\sigma = 80$	Proposed method	2.04×10^{-9}	1.37×10^{-9}	5.02×10^{-10}	4.68×10^{-10}
12.8	At $t_\sigma = 80$	Proposed method	9.16×10^{-11}	3.58×10^{-11}	1.78×10^{-11}	1.03×10^{-11}

(b) Spatial convergence for $La = 12\,000$ compared to results from the literature.

Ca		R/h			
		6.4	12.8	25.6	51.2
At $t_{cap} = 250$	Proposed method	5.42×10^{-7}	9.21×10^{-9}	1.13×10^{-9}	6.53×10^{-11}
	[42]	2.18×10^{-6}	2.32×10^{-7}	5.91×10^{-8}	–
	[12] Cartesian	1.44×10^{-6}	3.40×10^{-7}	5.00×10^{-8}	–
	[13]	6.68×10^{-6}	1.07×10^{-6}	1.20×10^{-7}	–
At $t_\sigma = 20$	Proposed method	1.52×10^{-9}	4.49×10^{-11}	1.72×10^{-12}	1.55×10^{-13}

are much smaller which will lead to reduced errors in the general case. We see in Figs. 14(a) and 14(b) the proportional link between the curvature's standard deviation and the spurious currents that arise mostly because of κ_σ. The normal deviation for this column case is, as expected, equal for κ_{LS} at orders 2 and 4 because of the regularity of the particular circular surface as it was shown in §4.2. When the standard deviation becomes stable after 4 half-oscillations, the surface oscillates one more time until parasitic currents are damped by viscosity and numerical equilibrium is reached.

The use of a fifth order WENO scheme for the advection of the level set does not change this convergence rate as expected (down to order $5 - 2 = 3$). We believe that this is due to the smoothness of the level set function and to the very small velocities, which introduce errors in ϕ much less than $O(h^5)$, added to the dynamics of the case that searches to minimize the geometrical errors. We show in §4.5 how the advection of the level set impacts the appearance of spurious currents.

4.3.3. Varying the Laplace number

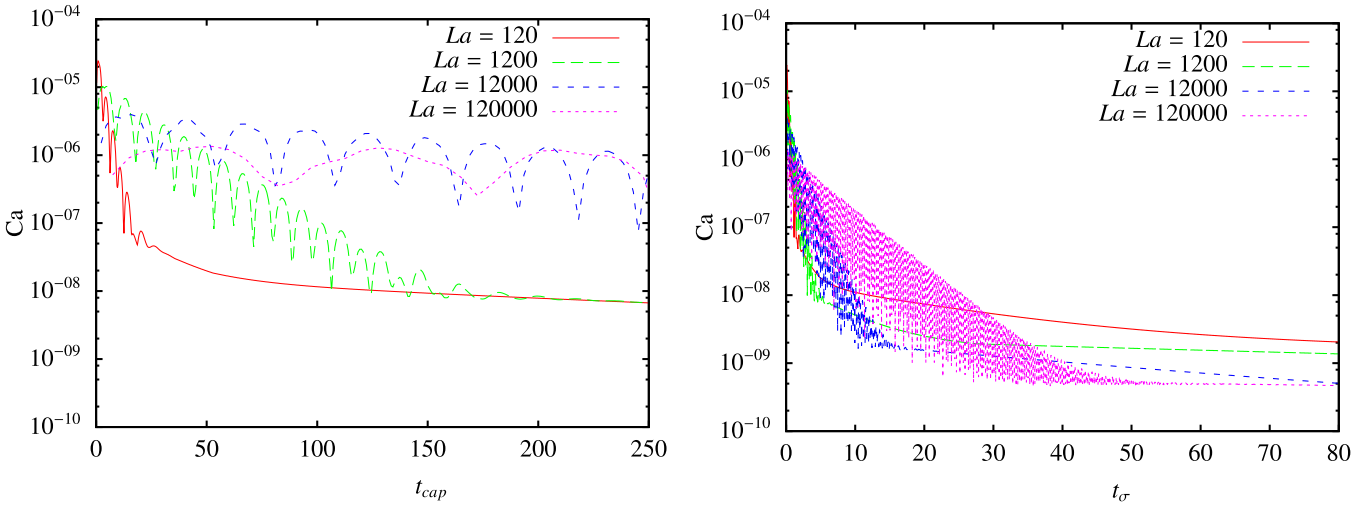
We study the effect of a varying the Laplace number by changing the surface tension coefficient σ at fixed density and viscosity. Augmenting La will make the interface response to numerical errors "stiffer" thus reaching a stable state after a longer time in the T_σ reference frame.

Here we used the $CP^2_\perp$ method with a 4th order κ_{LS} and the spatial discretization is $R/h = 6.4$. Because of the varying surface tension coefficient, the time step is adapted as $\Delta t = \{3, 1, 0.3, 0.1\} \times 10^{-4}$ respectively for $La = \{120, 1200, 12\,000, 120\,000\}$.

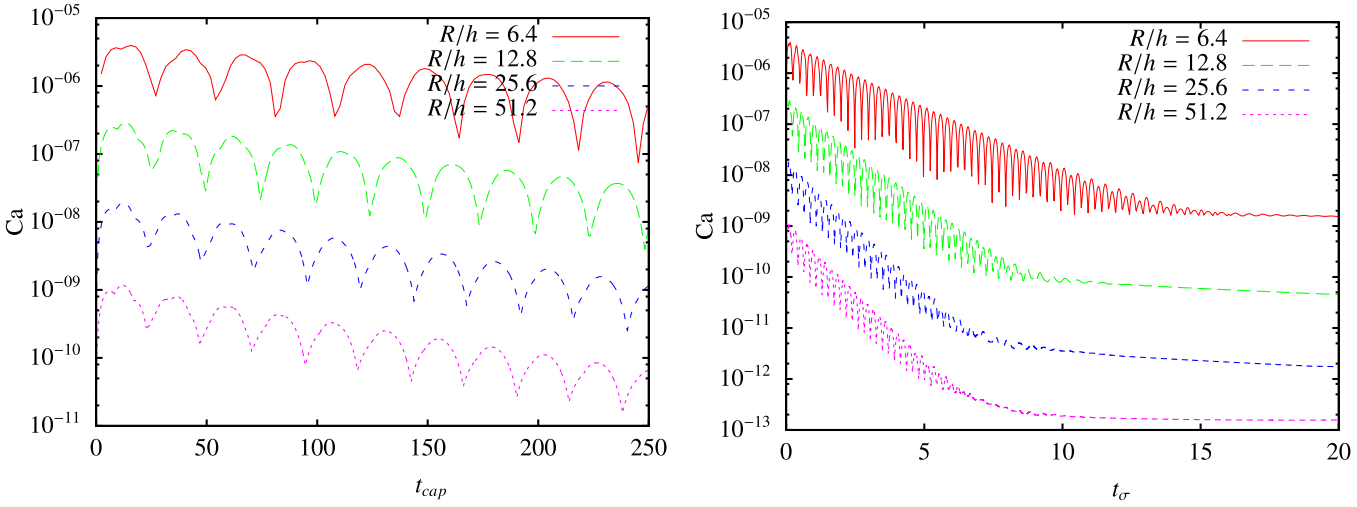
We have summarized and compared the numerical results to the ones from the literature in Table 6 where we can see that the spurious currents obtained are much smaller for the ratio $R/h = 6.4$. Moreover, the convergence rate with respect to h is also much higher. Note that in these references Ca is computed at $t_{cap} = t\sigma/(D\mu) = 250$, a time when the simulation has not converged to equilibrium velocity, which as we observed in our simulations (see Fig. 15(a)), is around $t_{cap} = 25\,000$ for $La = 120\,000$. Thus the velocity at that time is still oscillating significantly toward equilibrium and is not minimum as shown in Fig. 15(b). Consequently, we have given a new table of Ca number for different $t_\sigma = t/T_\sigma$ (when the velocity is stabilized), which we believe is more representative.

The results obtained for $R/h = 6.4$ suffer from excessively coarse spatial discretization as the regularization width is $\epsilon = 2h \simeq 0.3\,R$. Thus we propose to study the case where $R/h = 12.8$ for which we show more complete numerical results in Fig. 15, where Δt has been divided by 3. The results are qualitatively equivalent those obtained by [14] in a VOF framework for the same discretization. However, the velocity RMS is approximately 20 times smaller in our case thanks to the higher order of convergence of the curvature error of the $CP_\perp$ method. This difference will increase rapidly with finer discretization. As expected, increasing the Laplace number makes the interface stiffer and takes more time to reach equilibrium, where the oscillations observed converge toward an approximate period of $0.44\,T_\sigma$.

We observe in Figs. 16(c) and 16(d) a reduction of the curvature error measures when augmenting La. The Ca curve follows the standard and normal deviations showing the importance of those two criteria in the reduction of spurious currents. This is consistent with the idea that a higher surface tension coefficient makes the surface response stiffer, thus impacting the level set's movement in the vicinity of Γ making it numerically more homogeneous.



(a) Varying the Laplace number for $R/h = 6.4$.



(b) Varying the spatial discretization for $La = 12,000$.

Fig. 15. Static column case: convergence study using the $CP^2_\perp$ method and order 4 κ_{LS} calculation in the T_{cap} (left) and T_σ (right) reference frames.

4.4. Static viscous column equilibrium with variable density

In order to study how the method performs in general fluid simulations, we study the effect of surface tension on the column problem with a variable density.

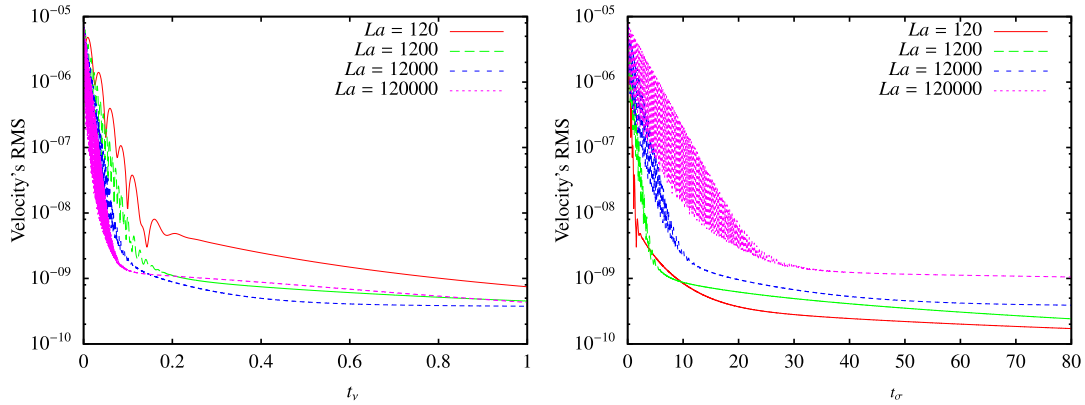
4.4.1. General configuration

We take exactly the same configuration as in §4.3 but change the density ρ_1 of the fluid inside the column while keeping the density outside ρ_2 fixed and σ constant. This has the effect of increasing the Laplace number. As ρ_2 does not change, the time step is restricted by this lower bound and we fixed $\Delta t = 1 \times 10^{-4}$. However, for increasing density, we observed in the first time steps a large error in the velocity field that we believe is due to the quick diffusion of the initial errors on the surface tension term by the reduced kinematic viscosity $\nu = \mu/\rho$. To counter this phenomenon, we reduced Δt to 1×10^{-6} for the first hundred computational steps, leaving time for the simulation to stabilize, and then smoothly increased it to 1×10^{-4}.

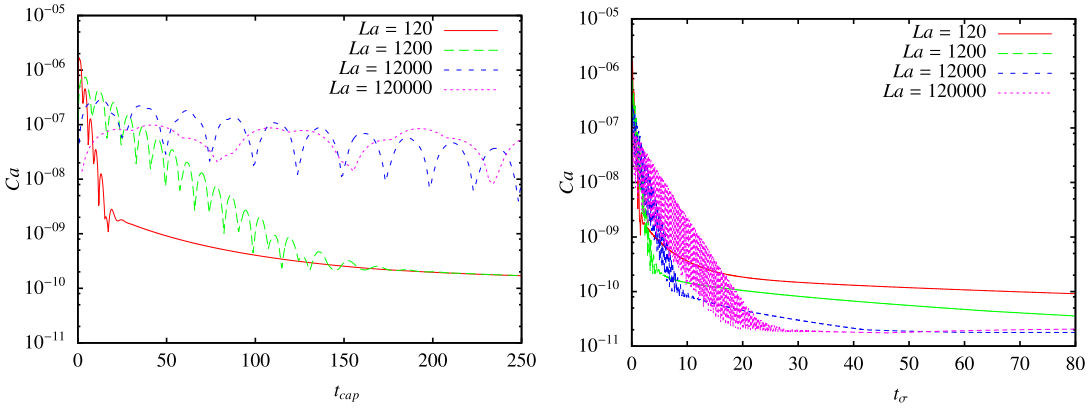
4.4.2. Varying the density

The surface tension coefficient is set to 300 while the density is varied as $\rho_1 = \{1, 10, 100, 1000\}$ with corresponding Laplace numbers $La = \{120, 660, 6060, 60\,060\}$, calculated as:

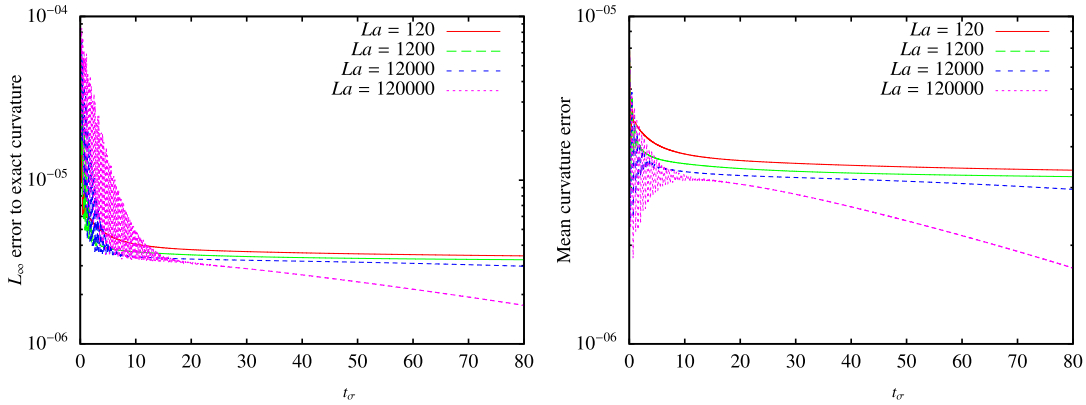
$$La_{\rho_1,\rho_2} = \frac{\sigma \overline{\rho} L}{\mu^2} \tag{40}$$



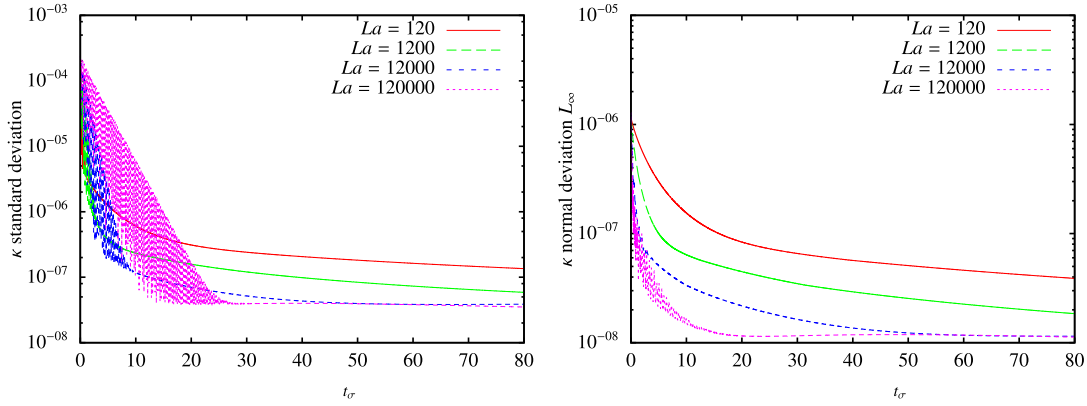
(a) Velocity's RMS for different time scales: T_ν and T_σ.



(b) Capillary number for different time scales: T_{cap} and T_σ.



(c) L_∞ error to exact curvature and error on the mean curvature.



(d) Standard deviation and L_∞ normal deviation.

Fig. 16. Static column case: convergence study by varying the Laplace number using the $CP^2_\perp$ method and order 4 κ_{LS} calculation with $R/h = 12.8$.

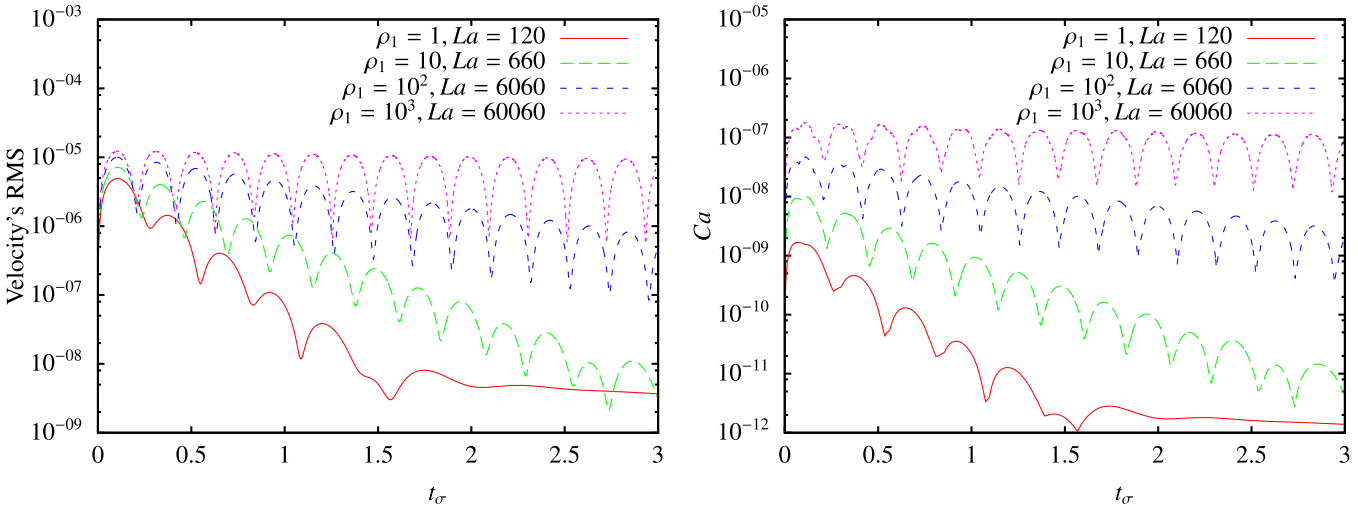


Fig. 17. Static column case with variable density: convergence study for the RMS of the velocity and Ca number by varying the density inside the column using the $CP^2_\perp$ method and order 4 κ_{LS} with $R/h = 12.8$.

with

$$\overline{\rho} = \frac{\rho_1 + \rho_2}{2}.$$

The reference scales T_σ and U_σ are defined with that same mean density.

Fig. 17 shows coherent results for the constant density test in §4.3 that demonstrates the stability and accuracy of the method in the presence of variable density.

4.5. Advected viscous column equilibrium

In these test cases, we study the impact of the errors due to the advection of the surface through the level set advection by an initial constant velocity field. Theoretically, in the reference frame of the transported column, the spurious currents should tend to zero. Due to the grid aliasing effect and advection errors we expect the interface to reach equilibrium with more difficulty, while still creating minimized spurious currents.

4.5.1. General configuration

The fluid parameters used are the same as in §4.3: $\rho = \mu = 1$ in the two phases. The domain is a rectangle of size $[1, 2]$ with symmetry conditions for the top and bottom boundaries, and periodic conditions in the horizontal direction. The column has a radius of $R = 0.2$. We initialized the x-velocity to $U_x = 2$ in the whole domain so that, ideally, the velocity field should remain $\mathbf{U} = (U_x, 0)$ through time. The spurious currents are measured in the column reference frame and calculated as:

$$v_{rms} = \sqrt{\int_\Omega |\mathbf{v}(\mathbf{x}) - \mathbf{U}|^2 d\mathbf{x}}$$

and

$$Ca = \frac{\mu |\mathbf{v} - \mathbf{U}|_{max}}{\sigma}.$$

All the velocities are given in the velocity scale for the advection problem:

$$U_{adv} = \sqrt{U_\sigma \cdot U_x}$$

as well as the corresponding time scale:

$$T_{adv} = D/U_x.$$

The CFL involved here will be much larger than the time step restriction due to surface tension: $\Delta t_{CFL} \gg \Delta t_\sigma$.

Table 7
Advected column case: numerical results for varying R/h and approximate order of convergence for the $CP^2_\perp$ method and order 4 κ_{LS} calculation. Mean measures are taken in the intervals $t_{adv} \in [0.06, 0.2]$ for $La = 1200$ and $t_{adv} \in [0.03, 0.1]$ for $La = 12\,000$, v_{rms} is given in the U_{adv} reference frame.

La	R/h	Mean v_{rms}	O	Mean Ca	O	max Ca	O	$\frac{\max Ca}{\text{mean } Ca}$
1200	6.4	7.83×10^{-5}	–	1.23×10^{-6}	–	1.05×10^{-5}	–	8.53
	12.8	2.02×10^{-6}	5.28	3.73×10^{-8}	5.04	7.57×10^{-7}	3.79	20.27
	25.6	6.94×10^{-8}	4.86	1.56×10^{-9}	4.58	4.87×10^{-8}	3.96	31.20
	51.2	3.65×10^{-9}	4.25	1.10×10^{-10}	3.83	3.07×10^{-9}	3.99	28.02
	102.4	1.17×10^{-10}	4.96	5.00×10^{-12}	4.46	1.91×10^{-10}	4.01	38.27
12 000	6.4	3.86×10^{-4}	–	1.10×10^{-6}	–	3.93×10^{-6}	–	3.56
	12.8	5.38×10^{-6}	6.16	1.55×10^{-8}	6.16	2.89×10^{-7}	3.77	18.66
	25.6	3.11×10^{-7}	4.11	9.90×10^{-10}	3.97	1.90×10^{-8}	3.92	19.23
	51.2	2.81×10^{-8}	3.47	1.02×10^{-10}	3.27	1.19×10^{-9}	4.00	11.60
	102.4	7.35×10^{-10}	5.26	3.02×10^{-12}	5.08	7.57×10^{-11}	3.97	25.08

4.5.2. Spatial convergence

We study the impact of the advection error of the level set coupled with the errors due to the surface representation and curvature calculation by varying the spatial discretization in the same manner as in §4.3. This case was first introduced by Popinet et al. in [14] in a VOF method with Height Function curvature framework. We set $\Delta t = 3 \times 10^{-6}$ for the $La = 1200$ simulations and $\Delta t = 1 \times 10^{-6}$ for $La = 12\,000$, except for the $R/h = 102.4$ discretization where we divided Δt by three to avoid instabilities.

As shown in Fig. 18, for early times, we observe the same numerical results as in §4.3, as the errors dominating here are due to the fact that the interface is not at equilibrium. After time $t_{adv} = 0.04$ (resp. $t_{adv} = 0.02$) for $La = 1200$ (resp. $La = 12\,000$) we clearly observe the effects of the errors on the advection perturbing the interface and the curvature in small oscillations around the equilibrium state. The period of these perturbations is of scale $1/h$ in the T_{adv} reference frame. This result is due to aliasing where the interface (i.e. the level set) suffers from approximately the same numerical discretization aliasing every time it has been translated by one cell. This observation coincides with the remarks given in [14]. Yet our method for advection and curvature calculation is more precise since the errors in the spurious currents are much smaller (by a factor around 3×10^{-4} for the $R/h = 12.8$ discretization), as shown in Fig. 18 and Table 7.

The simulations never reach a non-oscillating equilibrium, so we computed mean values for the velocity's RMS and Ca in the interval $t_{adv} = [0.06, 0.2]$ for $La = 1200$ and $t_{adv} = [0.03, 0.1]$ for $La = 12\,000$. The mean capillary number computed at equilibrium is much higher than the one for the static case. The ratio $\frac{\max Ca}{\text{mean } Ca}$ gives a good indication of how the method efficiently manages to reduce the spurious currents counteracting the errors constantly introduced into the level set. The curvature error measures presented in Fig. 18(c) were also computed as a mean in the interval $t_{adv} = [0.06, 0.2]$ for $La = 1200$. The orders of convergence are approximately equal to 4 for the L_∞ and standard deviation, and between 3.5 and 4 for the normal deviation, a reduction for the later that is due to the 5th order WENO advection scheme as expected by Proposition 1.

We observe in Fig. 18(a) two different regimes: the first one where the non-equilibrated geometric circle tends to find its equilibrium, followed by a second one where the errors in solving the ϕ transport equation become dominant and work against reaching the converged equilibrium as studied in §4.3. The peak capillary number converges at order 4 as observed previously in §14 while the mean v_{rms} and the mean Ca converge between the orders 4 and 5, which is notably higher than the results obtained for the static case in §4.3. We observe in Fig. 18(a) higher amplitudes for Ca in the transport error regime (compared to the initial geometry equilibrium search regime) for coarser resolutions where the grid aliasing is bigger and the order 2 Navier–Stokes errors are more present. Hence the errors on the velocities reduce even more as $h \to 0$, and converge faster toward the "optimum" reference values fixed by the static case results thus attaining an order of convergence higher than 4.

As shown in Fig. 18(c), the spatial convergences for the capillary number (similar to the RMS velocity) is at least order 4 which is 3 orders of magnitude better than the L_2 norm presented in [14]; their L_∞ errors do not show convergence while ours, through the Ca number, is better than fourth-order. This clearly demonstrates that the use of a high-order advection scheme coupled with high-order curvature extension is necessary to reduce the spurious currents in general flow simulations.

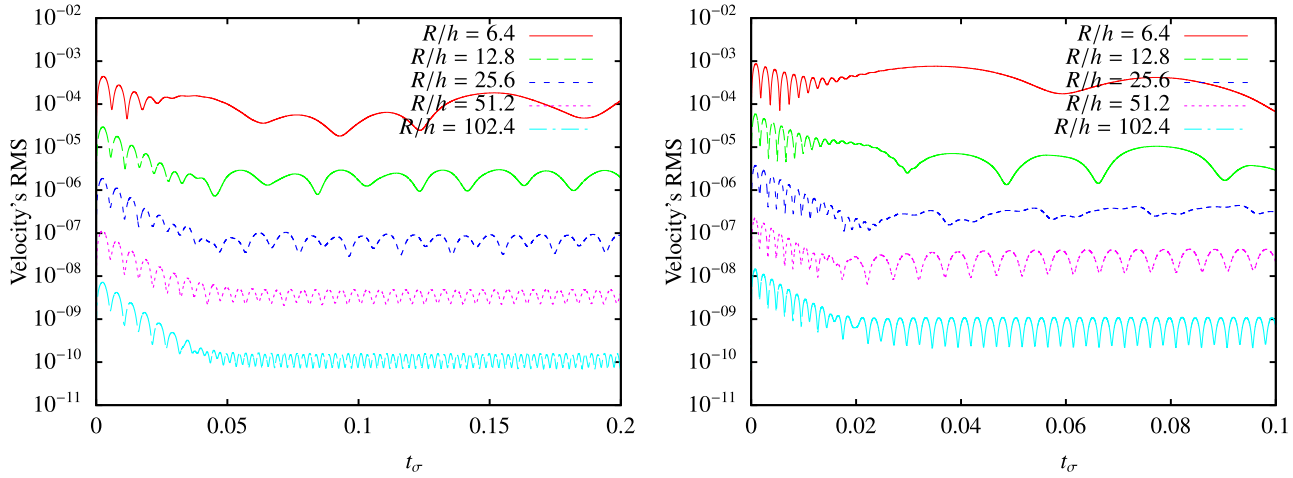
4.6. Zero gravity drop oscillation

In order to study the dynamic behavior of our method in the presence of non-constant curvature, we measured the oscillation period of a non-circular drop. The analytical results proposed in [43] were studied numerically in [12,4,44] and [45] in various frameworks.

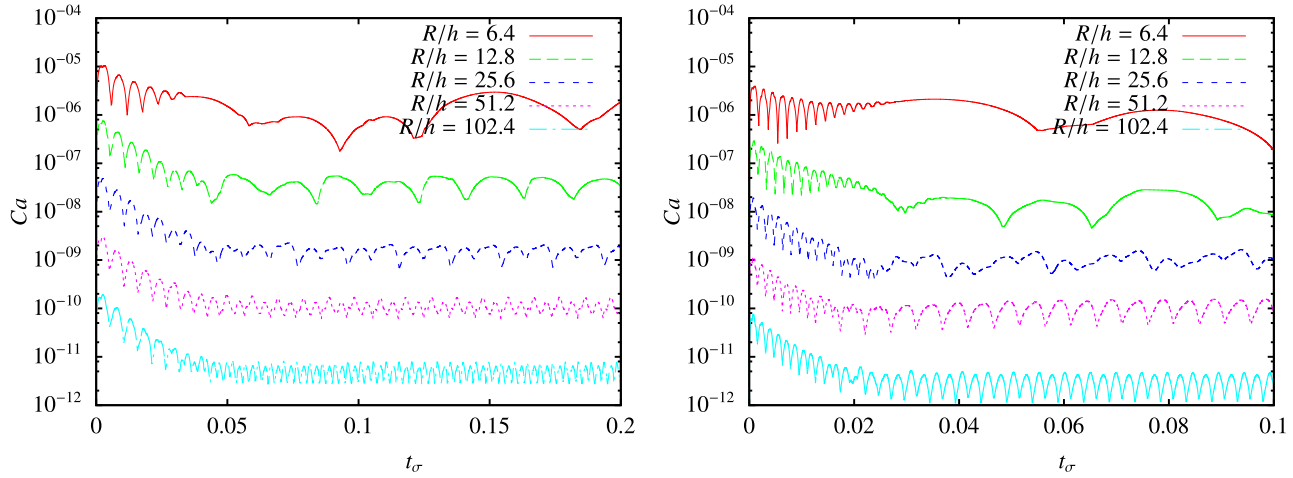
A 2D circular droplet is perturbed by a cosine function of mode n and amplitude A_0. The theoretical oscillation period ω as given by [43] is:

$$\omega^2 = \frac{n(n^2-1)\sigma}{(\rho_1+\rho_2)R_0^3}$$

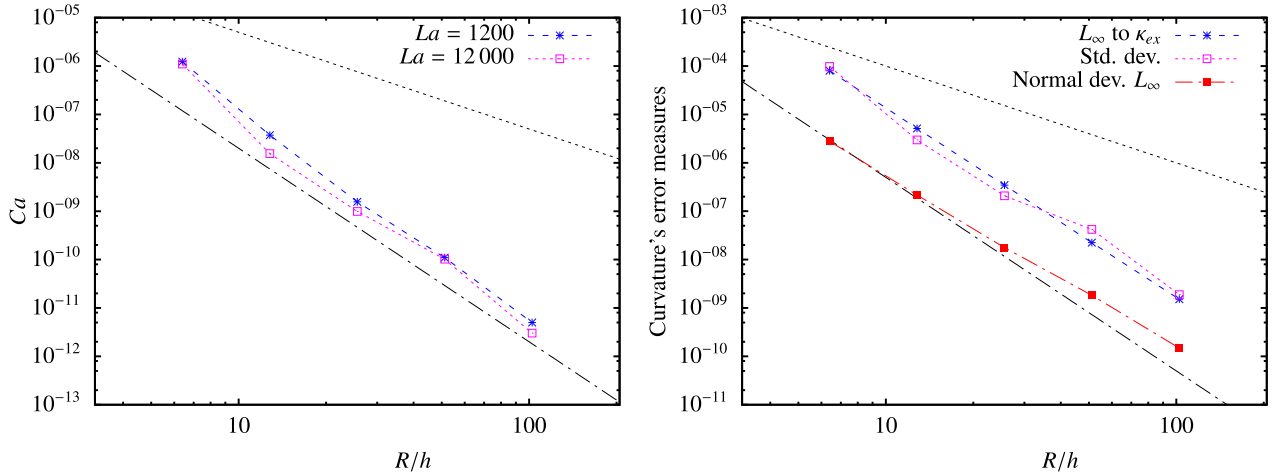
where ρ_1 (resp. ρ_2) is the density of the fluid inside the droplet (resp. outside).



(a) Velocity RMS for varying R/h, $La = 1200$ (left) and $La = 12\,000$ (right).



(b) Capillary number for varying R/h, $La = 1200$ (left) and $La = 12\,000$ (right).



(c) Left: mean Ca for $La = 1200$ and $La = 12\,000$. Right: mean curvature error measures for $La = 12\,000$. Spatial convergence with respect to R/h for mean values computed in the intevals $t_{adv} \in [0.06, 0.2]$ for $La = 1200$ and $t_{adv} \in [0.03, 0.1]$ for $La = 12\,000$.

Fig. 18. Advected column case: convergence study for the RMS velocity, Ca and curvature error measures for the $CP^2_\perp$ method and order 4 κ_{LS} computation, in the U_{adv} and T_{adv} reference frames.

Table 8
Spatial convergence of the oscillation error for the zero gravity drop oscillation, compared to the analytical solution from [43]. The value for T_{calc} at $R/h = \infty$ with our method has been computed with a Richardson's extrapolation.

E_ω	R/h					
	6.4	O	12.8	O	25.6	O
[12] Cartesian	4.04×10^{-2}	–	1.05×10^{-2}	1.94	0.37×10^{-2}	1.5
[45]	13.2×10^{-2}	–	6.1×10^{-2}	1.11	1.5×10^{-2}	2.02
Proposed method, 2nd order, eq. (31)	26.6×10^{-2}	–	8.30×10^{-2}	1.68	2.99×10^{-2}	1.47
Proposed method, 4th order, eq. (31)	6.57×10^{-2}	–	2.74×10^{-2}	1.26	0.54×10^{-2}	2.34
Proposed method, 4th order, eq. (32)	6.29×10^{-2}	–	2.68×10^{-2}	1.23	0.53×10^{-2}	2.33

T_{calc}	R/h				
	6.4	12.8	25.6	O	∞
Proposed method, 2nd order, eq. (31)	9.233	7.896	7.510	1.95	7.376
Proposed method, 4th order, eq. (31)	7.770	7.491	7.331	1.84	7.269
Proposed method, 4th order, eq. (32)	7.750	7.487	7.330	1.85	7.270
Analytical [43]	–	–	–	–	7.29138

4.6.1. General configuration

An initial column of radius $R_0 = 2$ is placed in the center of a $[-10, 10]$ square box with slip boundary conditions. The surface tension coefficient is fixed to $\sigma = 1$ and the physical parameters $\rho_1 = 1$, $\rho_2 = 0.01$, $\mu_1 = 0.01$ and $\mu_2 = 1 \cdot 10^{-4}$ which from eq. (40) corresponds to $La \approx 80\,000$. The column is perturbed by a cosine function which we translated in the level set representation as:

$$\phi(x) = |\mathbf{x}| - (R_0 + A_0 \cos(n\theta))$$

where we set $A_0 = 0.01\, R_0$ and $n = 2$. The time step was chosen as $\Delta_t = 3 \cdot 10^{-3}$.

4.6.2. Spatial convergence

Table 8 shows a comparison of the results in the literature with the results obtained by our method. The oscillation error is calculated as $E_\omega = \left| \frac{T_{calc}\omega}{2\pi} - 1 \right|$ with T_{calc} measured between the picks in the surface minimum x position, averaged over the 3 first periods. We show the impact of the precision of the Closest Point computation (with a corresponding threshold of $\epsilon = h^2$), the interpolation functions and κ_{LS} on the accuracy of the results which, for the most precise method, show an improvement by a factor of approximately 3 compared to [45] while the error is around 1.5 times higher compared to [12]. This difference can be explained by the sensitivity of these numerical results to the fluid solver and the different parameters, like for example the regularization parameter for the Heaviside function which is not given in [12] or the number of periods used for the measurements. We see an approximate second-order of convergence for the oscillation error.

As the oscillation error measures the dynamic behavior of the drop, we do not expect orders of convergence as high as the error on spurious currents. Indeed, in this more complex simulation, the errors are dominated by the flow solver which is restricted to second-order. However, we clearly observe on the error reduction the interest of computing the curvature extension up to a fourth-order accuracy. Particularly, the second-order method has higher error (3 to 5 times) compared to the fourth-order. Fig. 19 shows a better dynamical behavior when using the high order scheme, particularly for the low resolution $R/h = 6.4$ where the less accurate method leads to non-smooth oscillations with increasing amplitude. As expected, this demonstrates the clear advantage of using high order numerical methods, particularly when the curvature is large relatively to the spatial discretization.

Moreover, the theory is based on a linear regime study in an infinite fluid and so complex numerical conditions for the simulation may never converge to the analytical result but more toward a numerical equilibrium. The oscillation period extrapolated with Richardson's method on our measures is 7.270 which yields close to second-order spatial convergence, as expected by the fluid solver limitation.

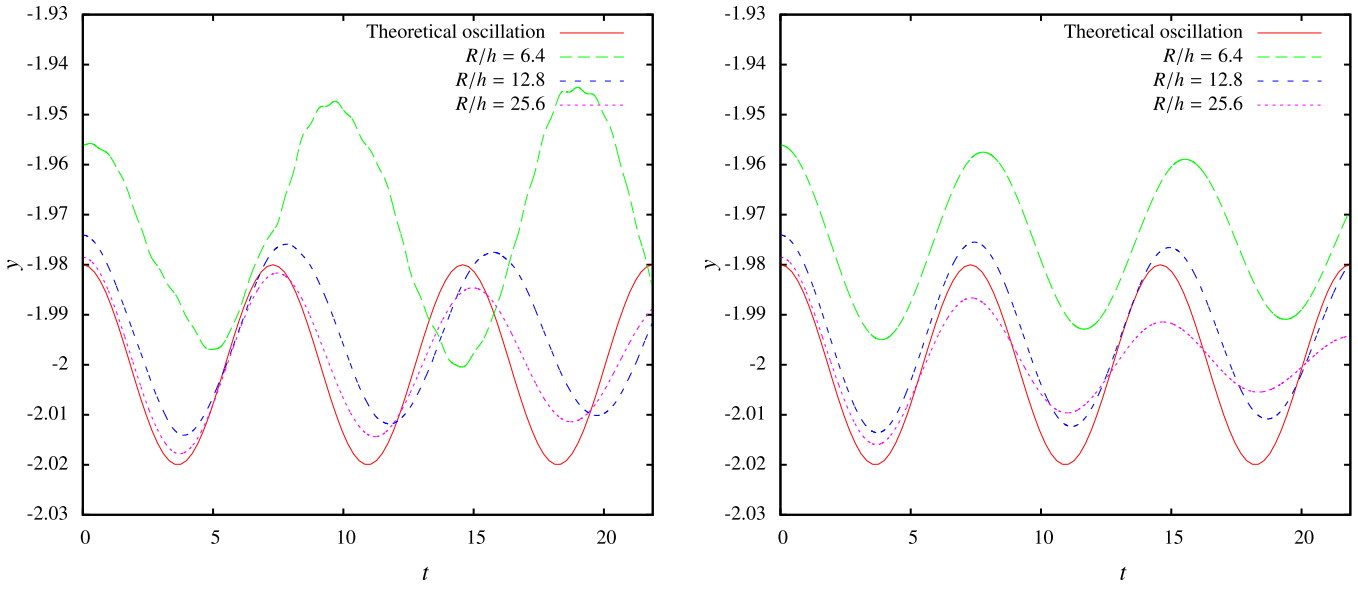
4.7. 2D bubble rise

4.7.1. General configuration

For this study we used the $CP^2_\perp$ algorithm and order 4 κ_{LS} calculation.

We consider the rise of a 2D bubble under the effect of buoyancy following [46,16] numerical studies. A disc of diameter $D = 0.5$ lies at position $[0.5, 0.5]$ at $t = 0$, inside a domain of size $[0, 0] \times [1, 2]$ with no-slip conditions on top and bottom walls and free-slip on left and right boundaries. The densities and viscosities inside (resp. outside) the bubble are ρ_2 and μ_2 (resp. ρ_1 and μ_1) and the gravity is set to 0.98.

The reference length is $L_{rise} = D = 0.5$ with an associated time scale of $T_{rise} = L_{rise}/U_{rise}$ and $U_{rise} = \sqrt{gD}$.



(a) Proposed method, 2^{nd} order, eq. 31 (b) Proposed method, 4^{th} order, eq. 32

Fig. 19. Spatial convergence of the oscillation for the zero gravity, compared to the analytical solution from [43], minimum height of the interface over time. The theoretical oscillation is plotted as reference, without the damping of the amplitude.

Table 9
2D bubble rise case: physical parameters.

Case	ρ_1	ρ_2	μ_1	μ_2	σ
A	1000	100	10	1	24.5
B	1000	1	10	0.1	1.96

We study the evolution of the center of mass of the bubble as numerically computed in the whole domain:

$$(x_c, y_c) = \frac{\sum \mathbf{x} H_\epsilon(\phi/|\nabla\phi|)}{\sum H_\epsilon(\phi/|\nabla\phi|)}.$$

The rising bubble velocity is computed in a similar fashion as:

$$(u_c, v_c) = \frac{\sum \mathbf{u} H_\epsilon(\phi/|\nabla\phi|)}{\sum H_\epsilon(\phi/|\nabla\phi|)}.$$

The circularity shows information on the deformation of the bubble surface and is written:

$$c = \frac{\pi D}{perimeter} \tag{41}$$

where the perimeter is numerically calculated with:

$$perimeter = h^2 \sum \delta_\epsilon(\phi/|\nabla\phi|). \tag{42}$$

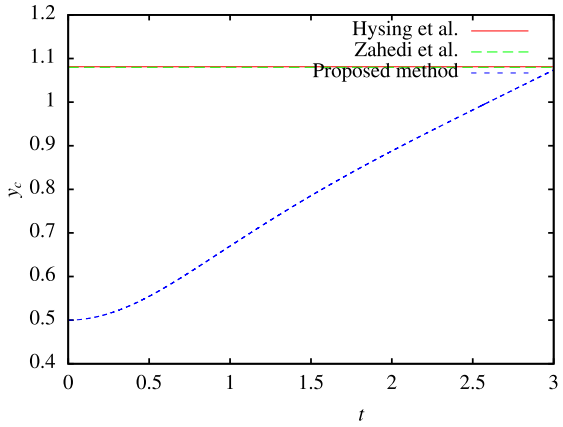
The time step used in all simulations is $\Delta t = 3 \times 10^{-4}$.

4.7.2. Spatial convergence

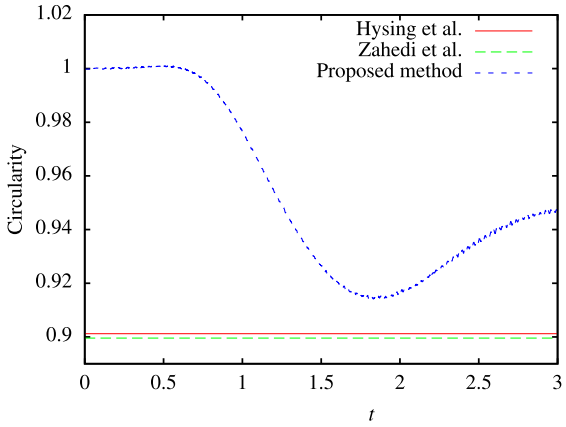
As seen in [16] for a finite element and level set framework, we propose to study the accuracy of the method in comparison to the numerical benchmark presented in [46] with various numerical methods. The numerical results compared in this test case have to be taken with care: they were obtained with multiple flow solvers that have different spatial and temporal discretization schemes. Moreover, the results in [46] presented here are obtained from extrapolation of numerical data, not theoretical nor experimental reference values.

The physical parameters of the two reference cases are presented in Table 9.

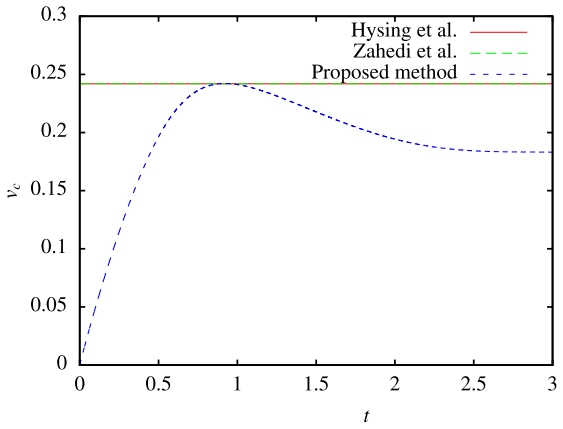
The numerical results for test case A are illustrated in Fig. 20 and summarized and compared to the literature in Table 10. They are in good agreement with the reference benchmark permitting us to validate our method for this rising bubble case. The maximum velocity of the bubble and the time it arises both fit the reference values and converge at the order 2. However, we observe a more important difference in the bubble circularity and center of mass between our results and



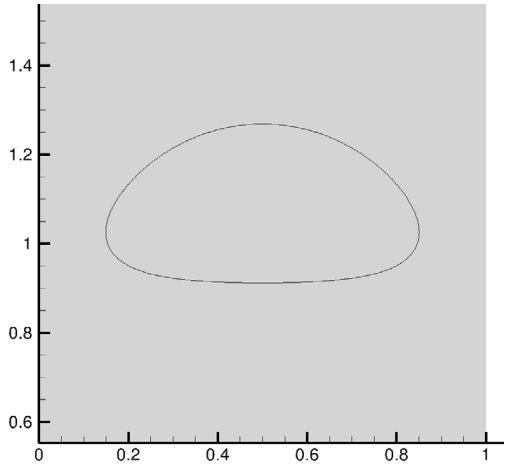
(a) Center of mass y_c.



(b) Circularity of the interface.



(c) Rise velocity v_c.



(d) Bubble's shape.

Fig. 20. 2D bubble rise case A: center of mass, circularity, rise velocity and bubble's shape at $t = 3$ for $R/h = 80$. Extremum results from [46,16] are drawn as lines.

Table 10
2D bubble rise case: numerical results compared to [46,16].

(a) Comparative results with $CP_\perp$ and order 4 κ_{LS}, for $R/h = 80$.

Measure	Proposed method	[16] coupled method	Ref. [46]
c_{min}	0.914	0.89955	0.9012 ± 0.0001
$t\|c = c_{min}$	1.817	1.909	1.9
$v_{c,max}$	0.2421	0.24190	0.2419 ± 0.0002
$t\|v = v_{c,max}$	0.9222	0.929	$0.921 \leq t \leq 0.932$
$y_c\|t = 3$	1.0761	1.07989	1.081 ± 0.001

(b) Results with the proposed method and the simple κ_{LS} method with no curvature extension, varying R/h. The order of convergence is computed by comparing the numerical results for the two finest discretizations to Richardson's extrapolation.

Measure	R/h				Richardson's extrapolation	O
	10	20	40	80		
Proposed method						
c_{min}	0.976	0.931	0.924	0.914	0.910	1.8
$t\|c = c_{min}$	1.769	1.934	1.806	1.817	1.826	1.2
$v_{c,max}$	0.2385	0.2411	0.2419	0.2421	0.2422	2.0
$t\|v = v_{c,max}$	0.9435	0.9282	0.9237	0.9222	0.9217	1.9
$y_c\|t = 3$	1.0587	1.0635	1.0708	1.0746	1.0761	1.8

the benchmark than in [16] for the same discretization. First, the circularity is computed in a level set framework from equations (41) and (42) which is as challenging as in a Eulerian framework it involves the regularized singular Dirac mass thus bringing oscillations because of the aliasing due to the grid. Therefore the computation of the value c_{min} and the measure of the time it takes to reach it is sensitive. We also observe a lower center of mass at $t = 3$.

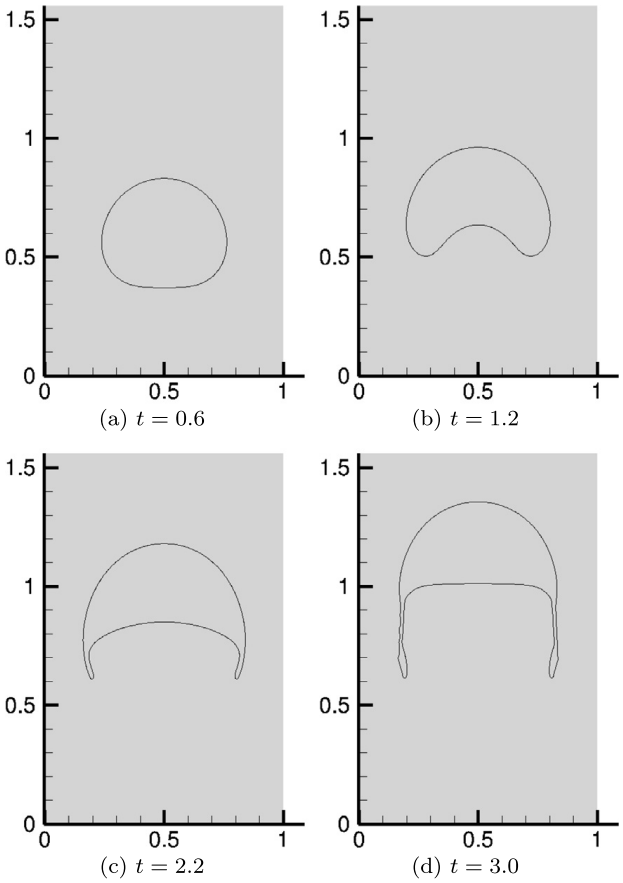


Fig. 21. 2D bubble rise case B: bubble shape for different times and for $R/h = 80$.

We believe that the numerical deviations come from the multiple differences in the numerical schemes and approximations used. We also believe that a better precision in the curvature computation at small scales with our method, as shown in §4.6, particularly on the bottom left and right corners where the curvature becomes high, helps to keep the bubble more circular showing more resistance to the vertical motion.

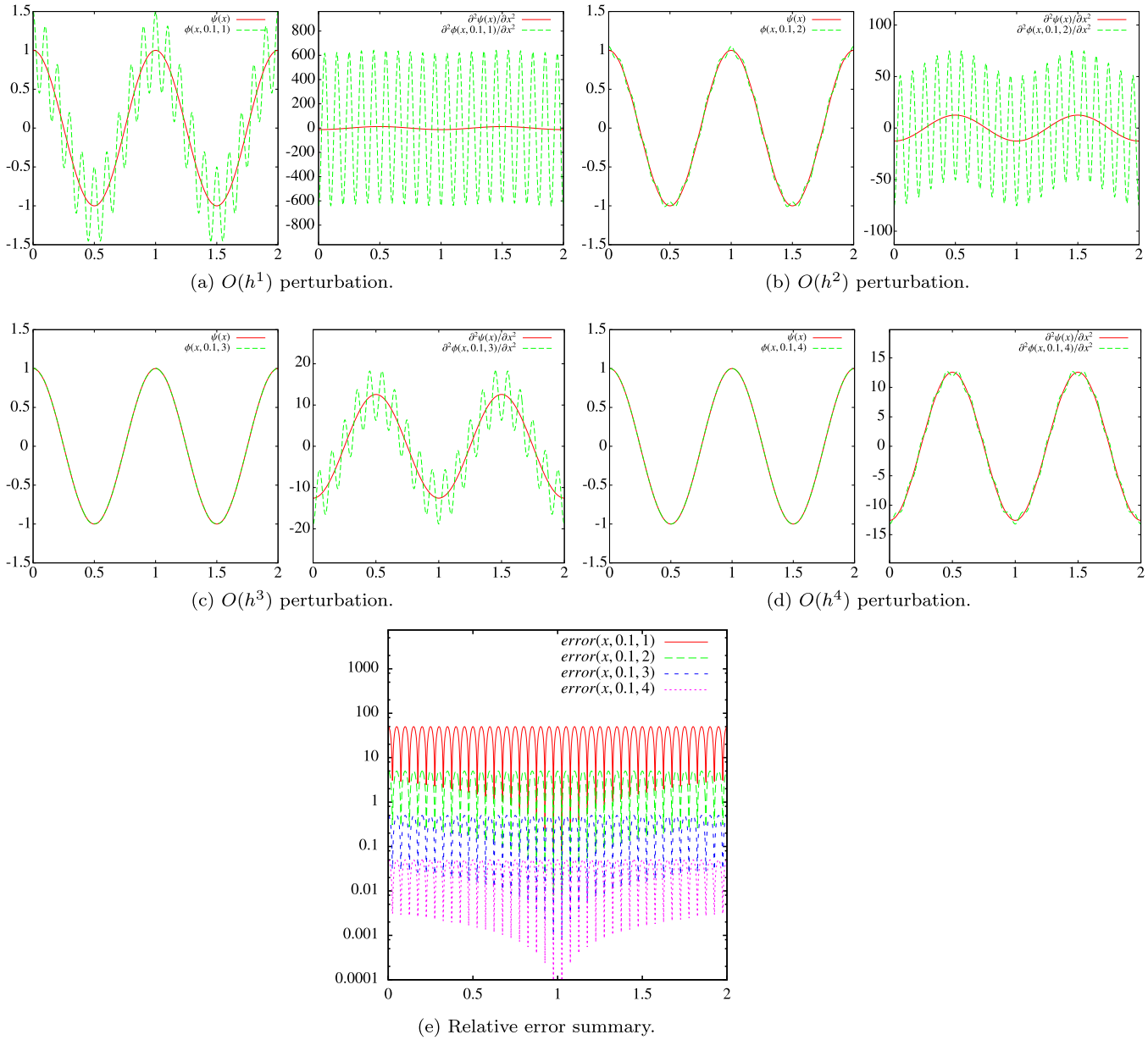
Fig. 21 shows qualitative results for test case B through the plots of the bubble's surface, simulated with a $R/h = 80$ discretization. In the early time-steps, $t < 2.2$, we see a similar behavior of this bubble to the one presented in [46]. After $t = 2.2$ we observe wider dragged filaments that resist reduction whereas in [46] the filaments disappear in the flow as residual small bubbles with the TP2D (finite elements and level set with reinitialization) method and are reduced to very thin filaments with MoonNMD (finite elements with a Lagrangian surface representation) and Fluent. We believe that the wider filaments we observe are due to:

1. The use of a regularized surface of width $\epsilon = 2h$ that smooths the blending zone and thus the dynamics of the bubble. This zone gets finer as $h \to 0$, though it does not disappear.
2. The more accurate computation of κ in high curvature areas, i.e. where the filaments arise, there is a more precise computation of the surface tension force preventing early decomposition of the surface.

5. Conclusion

In this article we present an improved method for the accurate computation of curvature in a level set framework. A complementary predicate (Criterion 2.3) concerning the normal deviation of the curvature in the CSF approach has been introduced and analyzed. We have studied the effect of perturbations on the surface's representation showing that it is necessary to use high-order transport techniques to calculate accurate and steady surface tension forces in a dynamic flow.

We illustrate the curvature calculation in a level set representation to demonstrate the necessity of an adequate curvature extension. The method proposed is based on a Closest Point algorithm improved with the colinearity criterion and reinterpolation. It yields fourth-order precision closest point approximation, enough to obtain fourth-order precision for the curvature error and standard deviation, and second-order precision for the normal error in general geometry. The algorithm is easy to implement, compared to Height Function methods in a VOF framework [18] and later works. It is also straightforward to extend to three dimensions while not being too computationally costly.



(a) $O(h^1)$ perturbation.

(b) $O(h^2)$ perturbation.

(c) $O(h^3)$ perturbation.

(d) $O(h^4)$ perturbation.

(e) Relative error summary.

Fig. 22. Second derivative errors in the presence of perturbations. Note that the y axis for the second derivatives plots have different scales.

We validate our approach on geometric, static, translating and complex dynamical cases in comparison to the literature. In particular we show that fourth-order accurate computation of curvature and surface tension forces can greatly reduce spurious currents even when the surface is transported, although the flow solver remains second-order accurate. We also observe good agreement of our numerical results with state-of-the-art complex simulations like the rise of a bubble.

Acknowledgements

This study was carried out with financial support from the French National Research Agency (ANR) in the framework of the "Investments for the future" Programme IdEx Bordeaux (ANR-10-IDEX-03-02), Cluster of excellence CPU.

The authors wish to thank the Aquitaine Regional Council for the financial support towards a 432-processor cluster investment, located in the I2M laboratory. Computer time for this study was also provided by the computing facilities MCIA (Mesocentre de Calcul Intensif Aquitain) of the Université de Bordeaux and of the Université de Pau et des Pays de l'Adour.

Appendix A

A.1. Second derivative error amplitude

In this paragraph we study the effect of numerical errors on derivatives through the introduction of perturbations in an analytical smooth function ψ. Let ψ, e and ϕ be three $\mathbb{R} \to \mathbb{R}$ functions defined as:

$$\psi(x) = \cos(2\pi x),$$
$$e(x,h,m) = h^m \cos(2\pi x/h),$$
$$\phi(x,h,m) = \cos(2\pi x) + h^m \cos(2\pi x/h),$$

as represented in Fig. 22. We can easily calculate the second derivative of ϕ:

$$\frac{\partial^2 \phi}{\partial x^2} = -4\pi^2(\sin(2\pi x) + h^{m-2}\sin(2\pi x/h))$$

which is equal to $\frac{\partial^2 \psi}{\partial x^2}$ perturbed at the scale h with an amplitude of the order h^{m-2}.

The error in the second derivative is measured by:

$$error(x,h,m) = \left| \frac{\partial^2\phi/\partial x^2 - \partial^2\psi/\partial x^2}{\partial^2\psi/\partial x^2} \right|$$

when $\partial^2\psi/\partial x^2 \neq 0$ and is plotted in Fig. 22 for different perturbations scales. We clearly see that perturbations bigger than h^2 lead to errors superior to 1 on the second derivative which is unpropitious in numerical computations.

A.2. Detailed numerical results

Table 11
Static column case: numerical results in function of R/h and order of convergence for the $CP^2_\perp$ method, order 2 and order 4 κ_{LS} calculation, $La = 120$.

(a) Velocity errors measures and pressure error.

κ_{LS}	R/h	Ca at $t = \Delta t$	O	Ca at $t_\sigma = 30$	O	max Ca	O	$E(\Delta p_{mean})$ at $t_\sigma = 30$	O
Order 2 scheme	6.4	2.02×10^{-6}	–	6.59×10^{-9}	–	4.83×10^{-5}	–	3.05×10^{-4}	–
	12.8	6.60×10^{-7}	1.62	1.96×10^{-10}	5.07	1.38×10^{-5}	1.81	6.71×10^{-5}	2.18
	25.6	1.82×10^{-7}	1.86	9.73×10^{-12}	4.33	3.60×10^{-6}	1.94	1.60×10^{-5}	2.07
	51.2	4.72×10^{-8}	1.95	4.09×10^{-13}	4.57	9.03×10^{-7}	1.99	3.96×10^{-6}	2.01
	102.4	1.19×10^{-8}	1.98	1.34×10^{-14}	4.93	2.27×10^{-7}	1.99	9.91×10^{-7}	2.00
Order 4 scheme	6.4	1.50×10^{-6}	–	5.26×10^{-9}	–	2.43×10^{-5}	–	2.63×10^{-5}	–
	12.8	1.30×10^{-7}	3.53	1.47×10^{-10}	5.16	1.67×10^{-6}	3.87	3.55×10^{-6}	2.89
	25.6	9.29×10^{-9}	3.81	7.58×10^{-12}	4.28	1.05×10^{-7}	3.98	1.75×10^{-7}	4.35
	51.2	6.20×10^{-10}	3.90	4.11×10^{-13}	4.21	6.47×10^{-9}	4.03	6.52×10^{-9}	4.74
	102.4	3.74×10^{-11}	4.05	2.08×10^{-14}	4.30	4.13×10^{-10}	3.97	4.24×10^{-10}	3.94

(b) Curvature error measures at $t_\sigma = 30$.

κ_{LS}	R/h	$\lvert\frac{\kappa_{mean}-\kappa_{ex}}{\kappa_{ex}}\rvert$	O	L_∞ curvature error	O	κ_σ	O	L_∞ normal dev.	O
Order 2 scheme	6.4	3.05×10^{-4}	–	3.10×10^{-4}	–	1.01×10^{-5}	–	2.65×10^{-6}	–
	12.8	6.71×10^{-5}	2.19	6.73×10^{-5}	2.20	4.60×10^{-7}	4.45	1.18×10^{-7}	4.49
	25.6	1.60×10^{-5}	2.07	1.60×10^{-5}	2.07	4.54×10^{-8}	3.34	1.14×10^{-8}	3.36
	51.2	3.96×10^{-6}	2.01	3.96×10^{-6}	2.01	2.45×10^{-9}	4.21	5.71×10^{-10}	4.32
	102.4	9.91×10^{-7}	2.00	9.91×10^{-7}	2.00	6.31×10^{-11}	5.28	1.68×10^{-11}	5.09
Order 4 scheme	6.4	2.66×10^{-5}	–	2.97×10^{-5}	–	6.92×10^{-6}	–	1.85×10^{-6}	–
	12.8	3.56×10^{-6}	2.90	3.66×10^{-6}	3.02	2.39×10^{-7}	4.86	6.52×10^{-8}	4.82
	25.6	1.75×10^{-7}	4.35	1.83×10^{-7}	4.32	1.80×10^{-8}	3.73	6.96×10^{-9}	3.23
	51.2	6.51×10^{-9}	4.74	6.90×10^{-9}	4.73	7.02×10^{-10}	4.68	3.02×10^{-10}	4.53
	102.4	4.24×10^{-10}	3.94	4.47×10^{-10}	3.95	2.30×10^{-11}	4.93	1.27×10^{-11}	4.57

References

[1] A. Yarin, D. Weiss, Impact of drops on solid surfaces: self-similar capillary waves, and splashing as a new type of kinematic discontinuity, J. Fluid Mech. 283 (1995) 141–173.
[2] D. Bhaga, M. Weber, Bubbles in viscous liquids: shapes, wakes and velocities, J. Fluid Mech. 105 (1981) 61–85.
[3] R. Clip, J. Grace, M. Weber, Bubbles, Drops, and Particles, Academic Press, 1978.
[4] G. Tryggvason, B. Bunner, O. Ebrat, W. Tauber, Computations of multiphase flows by a finite difference/front tracking method. I. Multi-fluid flows, in: Lecture Series – von Karman Institute For Fluid Dynamics, 1998, 7-7.
[5] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, S. Zaleski, Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows, J. Comput. Phys. 152 (2) (1999) 423–456.
[6] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1) (1994) 146–159.
[7] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (1) (2002) 83–116.

[8] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2) (2000) 301–337.
[9] G.-H. Cottet, J.-M. Etancelin, F. Pérignon, C. Picard, High order semi-Lagrangian particle methods for transport equations: numerical analysis and implementation issues, ESAIM: Math. Model. Numer. Anal. 48 (4) (2014) 1029–1060, http://dx.doi.org/10.1051/m2an/2014009, http://www.esaim-m2an.org/article_S0764583X14000090.
[10] J. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modeling surface tension, J. Comput. Phys. 100 (2) (1992) 335–354.
[11] M.M. Francois, S.J. Cummins, E.D. Dendy, D.B. Kothe, J.M. Sicilian, M.W. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, J. Comput. Phys. 213 (1) (2006) 141–173.
[12] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids, J. Comput. Phys. 227 (4) (2008) 2674–2706.
[13] S. Popinet, S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, Int. J. Numer. Methods Fluids 30 (6) (1999) 775–793.
[14] S. Popinet, An accurate adaptive solver for surface–tension–driven interfacial flows, J. Comput. Phys. 228 (16) (2009) 5838–5866.
[15] S. Shin, S. Abdel-Khalik, V. Daru, D. Juric, Accurate representation of surface tension using the level contour reconstruction method, J. Comput. Phys. 203 (2) (2005) 493–516.
[16] S. Zahedi, M. Kronbichler, G. Kreiss, Spurious currents in finite element based level set methods for two-phase flow, Int. J. Numer. Methods Fluids 69 (9) (2012) 1433–1456.
[17] F. Denner, B.G.M. van Wachem, Fully-coupled balanced-force VoF framework for arbitrary meshes with least-squares curvature evaluation from volume fractions, Numer. Heat Transf., Part B, Fundam. 65 (3) (2014) 218–255, http://dx.doi.org/10.1080/10407790.2013.849996.
[18] S.J. Cummins, M.M. Francois, D.B. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (6) (2005) 425–434.
[19] W. Aniszewski, T. Ménard, M. Marek, Volume of fluid (VoF) type advection methods in two-phase flow: a comparative study, Comput. Fluids 97 (2014) 52–73.
[20] V. Dyadechko, M. Shashkov, Reconstruction of multi-material interfaces from moment data, J. Comput. Phys. 227 (11) (2008) 5361–5384, http://dx.doi.org/10.1016/j.jcp.2007.12.029.
[21] M. Jemison, E. Loch, M. Sussman, M. Shashkov, M. Arienti, M. Ohta, Y. Wang, A coupled level set-moment of fluid method for incompressible two-phase flows, J. Sci. Comput. 54 (2–3) (2013) 454–491, http://dx.doi.org/10.1007/s10915-012-9614-7.
[22] C.B. Macdonald, S.J. Ruuth, Level set equations on surfaces via the closest point method, J. Sci. Comput. 35 (2–3) (2008) 219–240.
[23] G. Bornia, A. Cervone, S. Manservisi, R. Scardovelli, S. Zaleski, On the properties and limitations of the height function method in two-dimensional Cartesian geometry, J. Comput. Phys. 230 (4) (2011) 851–862.
[24] S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, S. Zaleski, Numerical integration of implicit functions for the initialization of the VoF function, Comput. Fluids 113 (2015) 42–52.
[25] Q. Zhang, On a family of unsplit advection algorithms for volume-of-fluid methods, SIAM J. Numer. Anal. 51 (5) (2013) 2822–2850.
[26] L.C. Evans, J. Spruck, et al., Motion of level sets by mean curvature I, J. Differ. Geom. 33 (3) (1991) 635–681.
[27] P. Liovic, M. Francois, M. Rudman, R. Manasseh, Efficient simulation of surface tension-dominated flows through enhanced interface geometry interrogation, J. Comput. Phys. 229 (19) (2010) 7520–7544.
[28] A. Poux, S. Glockner, M. Azaïez, Improvements on open and traction boundary conditions for Navier–Stokes time-splitting methods, J. Comput. Phys. 230 (10) (2011) 4011–4027.
[29] P. Lubin, S. Glockner, Numerical simulations of three-dimensional plunging breaking waves: generation and evolution of aerated vortex filaments, J. Fluid Mech. 767 (2015) 364–393, http://dx.doi.org/10.1017/jfm.2015.62, http://journals.cambridge.org/article_S0022112015000622.
[30] K. Goda, A multistep technique with implicit difference schemes for calculating two-or three-dimensional cavity flows, J. Comput. Phys. 30 (1) (1979) 76–95.
[31] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl. 23 (1) (2001) 15–41.
[32] P.R. Amestoy, A. Guermouche, J.-Y. L'Excellent, S. Pralet, Hybrid scheduling for the parallel solution of linear systems, Parallel Comput. 32 (2) (2006) 136–156.
[33] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1) (1996) 202–228.
[34] B. Engquist, A.-K. Tornberg, R. Tsai, Discretization of Dirac delta functions in level set methods, J. Comput. Phys. 207 (1) (2005) 28–51.
[35] F. Denner, B. van Wachem, On the convolution of fluid properties and surface force for interface capturing methods, Int. J. Multiph. Flow 54 (2013) 61–64, http://dx.doi.org/10.1016/j.ijmultiphaseflow.2013.03.004, http://www.scopus.com/inward/record.url?eid=2-s2.0-84876485278&partnerID=40&md5=3826186e24b1a52c1d56c3ed4210fe65.
[36] C. Min, F. Gibou, Robust second-order accurate discretizations of the multi-dimensional Heaviside and Dirac delta functions, J. Comput. Phys. 227 (22) (2008) 9686–9695.
[37] G.-H. Cottet, E. Maitre, A level set method for fluid–structure interactions with immersed surfaces, Math. Models Methods Appl. Sci. 16 (3) (2006) 415–438.
[38] S.J. Ruuth, B. Merriman, A simple embedding method for solving partial differential equations on surfaces, J. Comput. Phys. 227 (3) (2008) 1943–1961.
[39] F. Denner, D.R. van der Heul, G.T. Oud, M.M. Villar, A. da Silveira Neto, B.G. van Wachem, Comparative study of mass-conserving interface capturing frameworks for two-phase flows with surface tension, Int. J. Multiph. Flow 61 (2014) 37–47.
[40] D. Fuster, G. Agbaglah, C. Josserand, S. Popinet, S. Zaleski, Numerical simulation of droplets, bubbles and waves: state of the art, Fluid Dyn. Res. 41 (6) (2009) 065001.
[41] F. Denner, B. van Wachem, Numerical time-step restrictions as a result of capillary waves, J. Comput. Phys. 285 (2015) 24–40, http://dx.doi.org/10.1016/j.jcp.2015.01.021, http://www.scopus.com/inward/record.url?eid=2-s2.0-84925115207&partnerID=40&md5=0e733960a4f87c2deb45cbfffc0142d9.
[42] M. Owkes, O. Desjardins, A mesh-decoupled height function method for computing interface curvature, J. Comput. Phys. 281 (2015) 285–300, http://dx.doi.org/10.1016/j.jcp.2014.10.036, http://www.sciencedirect.com/science/article/pii/S0021999114007189.
[43] S.H. Lamb, Hydrodynamics, Dover Books, 1945.
[44] S.V. Shepel, B.L. Smith, On surface tension modelling using the level set method, Int. J. Numer. Methods Fluids 59 (2) (2009) 147–171.
[45] D. Torres, J. Brackbill, The point-set method: front-tracking without connectivity, J. Comput. Phys. 165 (2) (2000) 620–644.
[46] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, L. Tobiska, Quantitative benchmark computations of two-dimensional bubble dynamics, Int. J. Numer. Methods Fluids 60 (11) (2009) 1259–1288.

7 Moment-of-fluid analytic reconstruction on 2D Cartesian grids

Journal of Computational Physics 328 (2017) 131–139

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Moment-of-fluid analytic reconstruction on 2D Cartesian grids

Antoine Lemoine [a,*], Stéphane Glockner [b], Jérôme Breil [c]

[a] Univ. Bordeaux, I2M, UMR 5295, F-33400 Talence, France
[b] Bordeaux INP, I2M, UMR 5295, F-33400 Talence, France
[c] CEA CESTA, 15 Avenue des Sablières, CS 60001, 33116 Le Barp Cedex, France

ARTICLE INFO

Article history:
Received 17 December 2015
Received in revised form 13 July 2016
Accepted 5 October 2016
Available online 12 October 2016

Keywords:
Moment-of-Fluid
Interface reconstruction
Cartesian grid

ABSTRACT

Moment-of-Fluid (MoF) is a piecewise linear interface reconstruction method that tracks fluid through its volume fraction and centroid, which are deduced from the zeroth and first moments. We present a method that replaces the original minimization stage by an analytic reconstruction algorithm on bi-dimensional Cartesian grids. This algorithm provides accurate results for a lower computational cost than the original minimization algorithm. When more than two fluids are involved, this algorithm can be used coupled with the minimization algorithm. Although this paper deals with Cartesian grids, everything remains valid for any meshes that are made of rectangular cells.

1. Introduction

Moment-of-Fluid (MoF) [1–12] is a method to represent and reconstruct interfaces in multiphase flow simulation. This method is the apex of the Volume-of-Fluid (VoF) methods using piecewise linear interface reconstruction [13,14]. MoF represents the interfaces with the first two moments of any material subset ω in a polygonal cell Ω, namely the volume $M_0(\omega)$ and the first momentum $\boldsymbol{M}_1(\omega)$.

$$M_0(\omega) = \int_\omega d\boldsymbol{x} \qquad \boldsymbol{M}_1(\omega) = \int_\omega \boldsymbol{x} d\boldsymbol{x} \tag{1}$$

Sometimes it is more convenient to use their relative equivalent quantities, namely the volume fraction $\mu(\omega)$ and the centroid $\boldsymbol{x}_c(\omega)$.

$$\mu(\omega) = \frac{M_0(\omega)}{M_0(\Omega)} \qquad \boldsymbol{x}_c(\omega) = \frac{\boldsymbol{M}_1(\omega)}{M_0(\omega)} \tag{2}$$

MoF consists in finding a polygonal approximation ω^ℓ of a reference subset $\omega^\star$ (see Fig. 1). The part of the boundary $\Gamma^\ell = \partial\omega^\ell \setminus \partial\Omega$ is an affine approximation of the reference interface $\Gamma^\star = \partial\omega^\star \setminus \partial\Omega$. Furthermore, ω^ℓ verifies the following minimization problem:

$$\text{Find} \quad \omega^\ell = \underset{\omega^\ell}{\operatorname{argmin}} \, |\boldsymbol{x}_c(\omega^\ell) - \boldsymbol{x}_c(\omega^\star)|^2 \qquad \text{such that} \qquad M_0(\omega^\ell) = M_0(\omega^\star) \tag{3}$$

* Corresponding author.
E-mail addresses: antoine.lemoine@bordeaux-inp.fr (A. Lemoine), glockner@bordeaux-inp.fr (S. Glockner), jerome.breil@u-bordeaux.fr (J. Breil).

http://dx.doi.org/10.1016/j.jcp.2016.10.013
0021-9991/

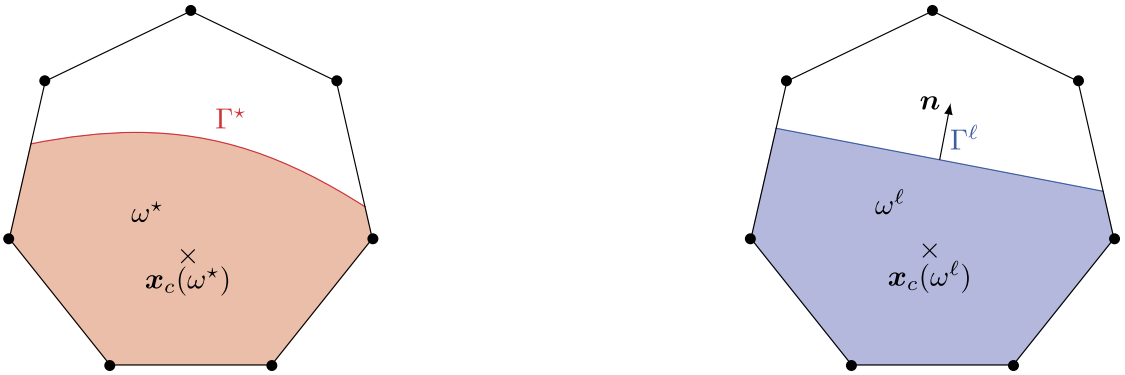


Fig. 1. Reference subset (left) and reconstructed subset (right).

In the remainder of this article, we use the following notations: the geometric elements with a $\star$ in exponent refer to the reference configuration, the geometric elements with an ℓ in exponent refer to the solution of problem (3) and the geometric elements without an exponent refer to any affine approximation.

Since the seminal publications [1–4], MoF has been adapted to specific applications. For instance, MoF has been exploited in an adaptive mesh refinement (AMR) context [5], used in arbitrary Lagrangian–Eulerian (ALE) schemes [6,8], coupled with level-set representation of multiphase flow [9], used in compressible multiphase flow [11], and adapted to axisymmetric coordinates [15] or in cylindrical geometry with an ALE approach [16]. Moreover, the accuracy of MoF has been improved thanks to the symmetric reconstruction [10] or with the introduction of filament capturing [12].

This article focuses on the implementation of MoF reconstruction on bi-dimensional Cartesian grids or any meshes composed of rectangular cells, which are widely used in computational fluid dynamics. While classic implementations use a time consuming minimization algorithm to solve the minimization problem, we propose a faster analytic reconstruction formula which takes advantage of the shape of the cells. The next section presents the demonstration of this analytic formula in two steps. First, we propose a parametrization of the locus of the centroids in a fixed volume. We show that this curve can be parametrized by four parabolas and four hyperbolas. Second, we find the minimal distance of the reference centroid to the curve with an orthogonal projection. This involves the computation of the minimal distance from a point to a hyperbola and to a parabola. In the remainder of this article, in section 3, we propose an analytical algorithm where we have reduced the search of the minimal distance to two parabolas and one hyperbola. In section 4, we discuss about existence and uniqueness of a solution. In section 5, we compare the proposed algorithm to the original minimization algorithm.

2. Analytic reconstruction

2.1. Description

Consider a rectangular cell of dimension (c_x, c_y) such as represented in the center of Fig. 2. The locus of the centroids $\boldsymbol{x}_c(\omega) := (g_x, g_y)$ for a given reference volume $V := M_0(\omega^\star)$ is a closed convex curve. In Fig. 2, we have represented the locus of the centroids for various reference volumes such that $V \leq 0.5 c_x c_y$. We observe 8 different configurations. 4 configurations where the reconstructed polygon is a triangle (odd numbers on the figure) and 4 configurations where the reconstructed polygon is a quadrangle (even numbers on the figure). Only the first two configurations can be considered since any other configuration can be transformed into the first two by symmetry and/or inverting the role of c_x and c_y. In paragraph §2.2, we prove that when the reconstructed polygon is a triangle, the locus is a hyperbola $\mathcal{H}$ and when the reconstructed polygon is a quadrangle, the locus is a parabola $\mathcal{P}$. When $V > 0.5 c_x c_y$, the method can be applied on the dual (or complementary) configuration $\widetilde{\omega}^\star := \Omega \setminus \omega^\star$, that is, we consider the volume $M_0(\widetilde{\omega}^\star) = M_0(\Omega) - V$ and the first momentum $\boldsymbol{M}_1(\widetilde{\omega}^\star) = \boldsymbol{M}_1(\Omega) - \boldsymbol{M}_1(\omega^\star)$.

2.2. Parametrization

The reconstructed line segment Γ can be defined with two parameters, namely the interface normal (n_x, n_y) and the distance to the origin ξ, that is:

$$\Gamma = \{(x, y) \in \Omega \,/\, n_x x + n_y y = \xi\} \tag{4}$$

In the triangle configuration, Γ intersects the bottom edge of the cell in $0 < \alpha \leq c_x$ and the left edge of the cell in $0 < \beta \leq c_y$ (see Fig. 3). For a given reference volume V and a normal (n_x, n_y), the intersection coordinates are given by:

$$\alpha = \sqrt{2V \frac{n_y}{n_x}} \qquad \beta = \sqrt{2V \frac{n_x}{n_y}} \tag{5}$$

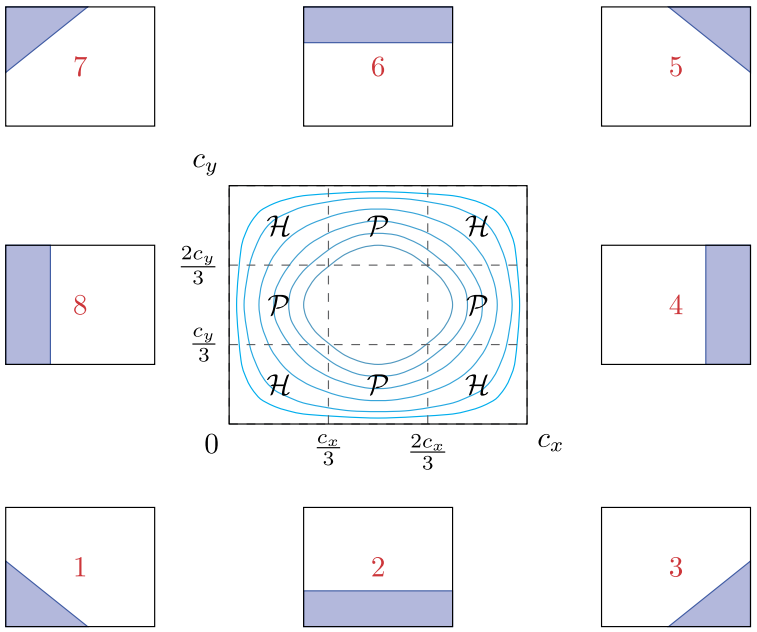


Fig. 2. Representation of the different configurations of the locus of the centroids (cyan) in a rectangular cell of dimension (c_x, c_y) for various reference volumes V such that $V \leq 0.5 c_x c_y$. Depending on the reconstructed polygon, the locus is either a hyperbola $\mathcal{H}$ or a parabola $\mathcal{P}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

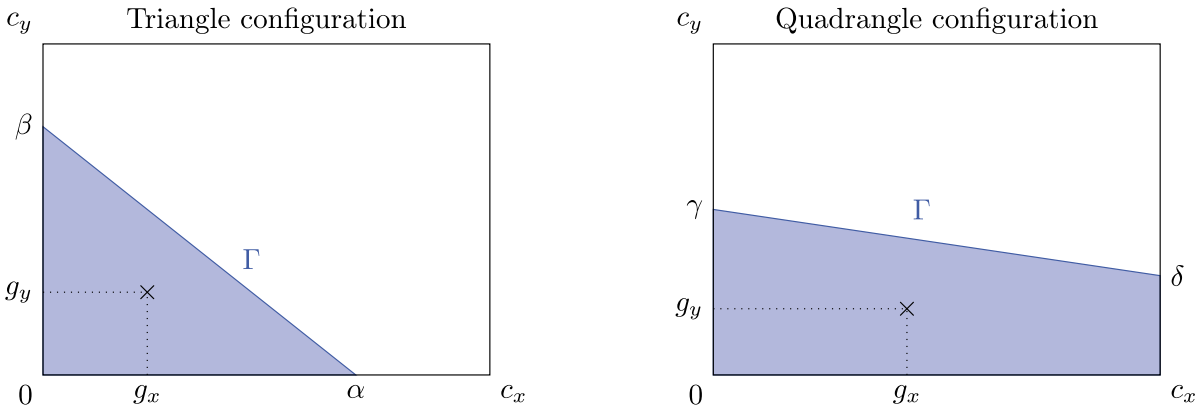


Fig. 3. Parametrization of triangle and quadrangle configurations.

The coordinates of the triangle centroid are given by:

$$g_x = \frac{1}{3}\sqrt{2V\frac{n_y}{n_x}} \qquad g_y = \frac{1}{3}\sqrt{2V\frac{n_x}{n_y}} \tag{6}$$

To express g_y as a function of g_x, we substitute n_x/n_y which gives an equation of a hyperbola:

$$g_y := \mathcal{H}(g_x) = \frac{2V}{9}\frac{1}{g_x} \tag{7}$$

The validity domain of this formula is given for the limit cases when $\beta = c_y$ and when $\alpha = c_x$. Simple calculations give:

$$g_x \in \left[\frac{2V}{3c_x}, \frac{c_x}{3}\right] \tag{8}$$

In the quadrangle configuration, Γ intersects the left edge of the cell in $0 < \gamma \leq c_y$ and the right edge of the cell in $0 < \delta \leq c_y$ (see Fig. 3). For a given reference volume V and a normal (n_x, n_y), the intersection coordinates are given by:

$$\gamma = \frac{V}{c_x} + \frac{n_x c_x}{2n_y} \qquad \delta = \frac{V}{c_x} - \frac{n_x c_x}{2n_y} \tag{9}$$

The coordinates of the quadrangle centroid are given by:

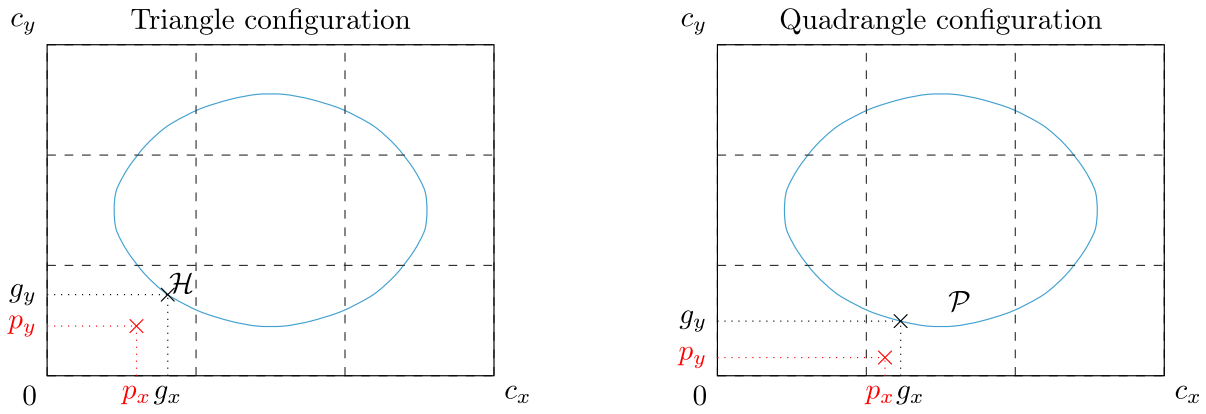


Fig. 4. Find the closest point (g_x, g_y) of a point (p_x, p_y) to the locus of the centroid in triangle and quadrangle configurations.

$$g_x = \left(\frac{1}{2} - \frac{c_x^2}{12V}\frac{n_x}{n_y}\right)c_x \qquad g_y = \left(\frac{V}{2c_x^2} + \frac{c_x^2}{24V}\frac{n_x^2}{n_y^2}\right)c_x \tag{10}$$

To express g_y as a function of g_x, we substitute n_x/n_y which gives an equation of a parabola:

$$g_y := \mathcal{P}(g_x) = \frac{V}{2c_x} + \frac{6V}{c_x}\left(\frac{1}{2} - \frac{g_x}{c_x}\right)^2 \tag{11}$$

The validity domain of this formula is given when the curve is not a hyperbola, *i.e.*:

$$g_x \in \left[\frac{c_x}{3}, \frac{2c_x}{3}\right] \tag{12}$$

Note that it is easy to check that the global curve defined by the union of all the parabolas and hyperbolas is three times differentiable, but the third derivative is not continuous.

We have shown that the locus of the centroid can be parametrized by four parabolas and four hyperbolas. In the next paragraph, we present how to find the minimal distance from the reference centroid to the various parts of the curve to find the global minimum.

2.3. Minimal distance

The minimal distance from any point $(p_x, p_y) \in \mathbb{R}^2$ to the locus of the centroid is its closest orthogonal projection (see Fig. 4). Since the curve is defined by parts, we need to express the minimal distance from any point to each part. The orthogonal projection $(g_x, g_y) = (g_x, \mathcal{H}(g_x))$ of the point (p_x, p_y) on the hyperbola $\mathcal{H}$ verifies the following equation.

$$(g_x - p_x, \mathcal{H}(g_x) - p_y) \cdot (1, \mathcal{H}'(g_x)) = 0 \tag{13}$$

This equation is the expression of the orthogonality of the vector defined by the point and its orthogonal projection with the tangent to the curve. Replacing $\mathcal{H}$ by its value (7) reveals a quartic equation on the coordinate g_x such that one of the real roots is the closest orthogonal projection.

$$g_x^4 - p_x g_x^3 + \frac{2}{9}V p_y g_x - \left(\frac{2V}{9}\right)^2 = 0 \tag{14}$$

The same process can be applied to find the orthogonal projection $(g_x, g_y) = (g_x, \mathcal{P}(g_x))$ of the point (p_x, p_y) on the parabola $\mathcal{P}$. The orthogonal projection verifies the following equation.

$$(g_x - p_x, \mathcal{P}(g_x) - p_y) \cdot (1, \mathcal{P}'(g_x)) = 0 \tag{15}$$

Substituting $\mathcal{P}$ by its expression (11) unveils a cubic equation on g_x.

$$g_x - p_x - \frac{12V}{c_x^2}\left(\frac{1}{2} - \frac{g_x}{c_x}\right)\left(\frac{V}{2c_x} - p_y\right) - \frac{72V^2}{c_x^3}\left(\frac{1}{2} - \frac{g_x}{c_x}\right)^3 = 0 \tag{16}$$

The dependence of equations (14) and (16) on so many parameters (p_x, p_y, c_x, c_y and V) makes it tricky to find the root that gives the closest orthogonal projection. The simplest way to find it is to compute all the roots and eliminate the roots outside their defined domain and find the one that gives the closest orthogonal projection. In our implementation, we use the algorithms proposed in [17] and [18] to accurately compute the cubic and quartic roots.

In the following paragraph, we define the transformations to solve the problem on any configuration.

2.4. Generalization to any configuration

As mentioned in paragraph §2.1, we have only considered a "reference configuration" where $V \leq 0.5c_xc_y$ and where the reference centroid belongs to one of the two first configurations (1 and 2) represented on Fig. 2. Here, we provide the formulas to transform any configuration into one of these two reference configurations and to transform the results back to the original configuration. In the following, we denote by $\boldsymbol{x}_c(\omega^\star) = (p_x, p_y)$ the reference centroid. All the coordinates are expressed in the cell referential where the origin corresponds to the bottom left corner.

We consider that we have two algorithms to handle problems 1 and 2 that take the cell dimensions (c_x, c_y) and the coordinates of the reference centroid (p_x, p_y) as input parameters. The output data are the coordinates of the normal (n_x, n_y). Using some transformations, it is possible to use this algorithm to solve any other problems.

If the reference volume is greater than the half of the cell volume $V > 0.5c_xc_y$, consider the dual volume $M_0(\widetilde{\omega}^\star) = c_xc_y - V$ and the dual reference centroid $\boldsymbol{x}_c(\widetilde{\omega}^\star) = (c_xc_y(c_x/2, c_y/2) - V(p_x, p_y))/(c_xc_y - V)$.

For problems 3 to 8, use the following transformations for input parameters and output data:

- Problems 3 and 4: cell dimensions (c_y, c_x), centroid coordinates $(p_y, c_x - p_x)$, normal $(-n_y, n_x)$.
- Problems 5 and 6: cell dimensions (c_x, c_y), centroid coordinates $(c_x - p_x, c_y - p_y)$, normal $(-n_x, -n_y)$.
- Problems 7 and 8: cell dimensions (c_y, c_x), centroid coordinates $(c_y - p_y, c_x)$, normal $(n_y, -n_x)$.

In the next section, we use the above preliminary work to propose an algorithm that replaces the original minimization algorithm to find the minimal distance from the reference centroid to the locus of the centroids.

3. Algorithm

The following algorithm computes the closest point $\boldsymbol{x}_c(\omega^\ell) = (g_x, g_y)$ of any reference centroid $\boldsymbol{x}_c(\omega^\star) = (p_x, p_y)$ on the locus of the centroids for a given volume V.

Step 1. If the volume of fluid is greater than the half of the cell volume $V > 0.5c_xc_y$, consider the dual volume $M_0(\widetilde{\omega}^\star) = c_xc_y - V$ and the dual reference centroid $\boldsymbol{x}_c(\widetilde{\omega}^\star)$.
Step 2. Find the subset of the plane where the reference centroid is located. According to the results listed below, consider the corresponding problems listed on Fig. 2:
- if $p_x \leq 0.5c_x$ and $p_y \leq 0.5c_y$ consider the problems 1, 2 and 8;
- if $p_x > 0.5c_x$ and $p_y \leq 0.5c_y$ consider the problems 2, 3 and 4;
- if $p_x > 0.5c_x$ and $p_y > 0.5c_y$ consider the problems 4, 5 and 6;
- if $p_x \leq 0.5c_x$ and $p_y > 0.5c_y$ consider the problems 6, 7 and 8.

Step 3. Solve one quartic (14) and two cubic (16) equations of the considered problems.
Step 4. Eliminate every root that gives a result which is outside the domain defined in equations (8) and (12).
Step 5. Find the root that gives the closest orthogonal projection.
Step 6. If the dual problem was solved, transform the result back to the primal problem.

Note that the third step requires performing the transformations defined in §2.4 in order to apply the formula (14) and (16).

After finding the coordinates of the centroid $\boldsymbol{x}_c(\omega^\ell)$, formula (6) and (10) can be used to obtain the normal (n_x, n_y) or any other quantity to represent the reconstructed interface.

4. Discussion on existence and uniqueness

In article [1], the authors address a particular attention to uniqueness and existence of a solution. The method proposed in the present article solves the same minimization problem (3) as defined in [1]. As a consequence, the results about uniqueness and existence of [1] are also valid for the proposed method, that is, the solution of MoF interface reconstruction exists and is unique except for a set of reference centroids that have zero area. Thus, a particular attention must be paid to how multiple solutions are handled and how the solutions are eliminated in the fourth step of the proposed algorithm.

To the question about how to handle multiple solutions, the answer is implementation dependent. The proposed algorithm consists in computing all the valid solution for every selected configurations and selects the solution that gives the closest distance to the reference centroid. In our implementation, if there are two solutions that give the same closest distance, only the first solution is kept. As regards the minimization, the choice between multiple solution depends on the initial condition of the minimization algorithm. Therefore, in both algorithms, the choice is in some ways arbitrary.

To the question about how to eliminate solutions that are outside their definition domain, a particular attention must be paid when the solutions are close to the border of their definition domain. It is possible due to the finite precision and numerical errors that a correct solution is eliminated because it lies out, but very close, to the border of its definition domain. A possible answer to this problem consists in giving a thickness to the border. That corresponds to accept the solutions that are outside their definition domain, but close to the border with a given epsilon.

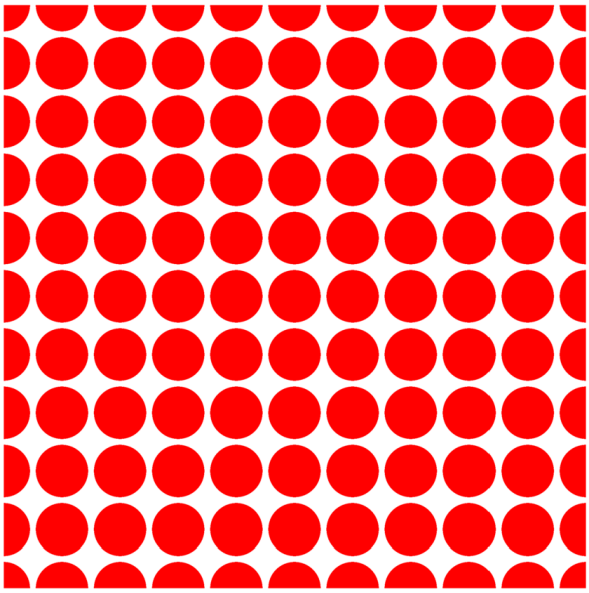


Fig. 5. MoF reconstruction of the static test case on a 2048^2 Cartesian grid.

Table 1
Time ratio minimization/analytic for the static test case on a 2048^2 Cartesian grid.

	Min. 10^{-15}	Min. 10^{-12}	Min. 10^{-9}	Min. 10^{-6}
Ana.	2.59	2.23	1.87	1.44

5. Test cases

We have implemented the proposed algorithm in the open-source code Notus (http://notus-cfd.org). We have tested this implementation on various cases from exact reconstruction to advected cases. We have compared the computational time with the standard minimization algorithm [1]. We consistently observe better performances for the analytic reconstruction, which span from 20% to 300% faster than the minimization method. This large range of performance results from the dependency of the method on a lot of parameters such as the tolerance value, the shape of the interface, the number of cells, and the number of fluids.

We have selected two test cases to provide a sample of the performance of this method. The first case is a static reconstruction on a shape that can be exactly reconstructed, *i.e.* the reference centroid belongs to the locus of the centroids. The second case is dynamic. It is closer to a practical case and will provide a better hint of the true performance than the static case.

In our implementation of the minimization algorithm, we use the line search algorithm [19] as recommended by [2]. The speed of the minimization algorithm depends on the Flood Algorithm used to reconstruct the interface from the normal and the volume fraction. In our implementation, we use the Flood Algorithm presented in [20] which is similar to the one presented in [1] but behaves better when two sections are very close to each other. The reader can refer to Appendix A for implementation details.

Remark. The minimization algorithm stops when the error reaches a prescribed tolerance value. Since the algorithm consists in bracketing the angle, the error is defined as the difference between the upper and lower bound of the bracketing. Note that this error should not be confused with the centroid defect. In other words, even if the centroid defect can not be zero, the bracketing error can always be as close as possible to zero, up to the machine error.

5.1. Static reconstruction

The static case consists in the reconstruction of many disks of radius 0.045 in a unit square domain $[0, 1]^2$ such as represented on Fig. 5. The disks are discretized such that an exact reconstruction is possible; the circle is approximated by straight line segments in the cells intersected by the interface. We use a Cartesian grid composed of 2048^2 cells to ensure a substantial computational time.

Table 1 presents the time ratio between minimization and analytic reconstruction for tolerance values 10^{-15}, 10^{-12}, 10^{-9} and 10^{-6}. These tolerance values are based on the angle, as defined in the previous remark.

In our experiments, we have observed that the centroid defect $|\boldsymbol{M}_1(\omega^\ell) - \boldsymbol{M}_1(\omega^\star)|$ given by the analytic reconstruction is on the order of the machine precision (10^{-16}). If we compare the computational time of the minimization process with

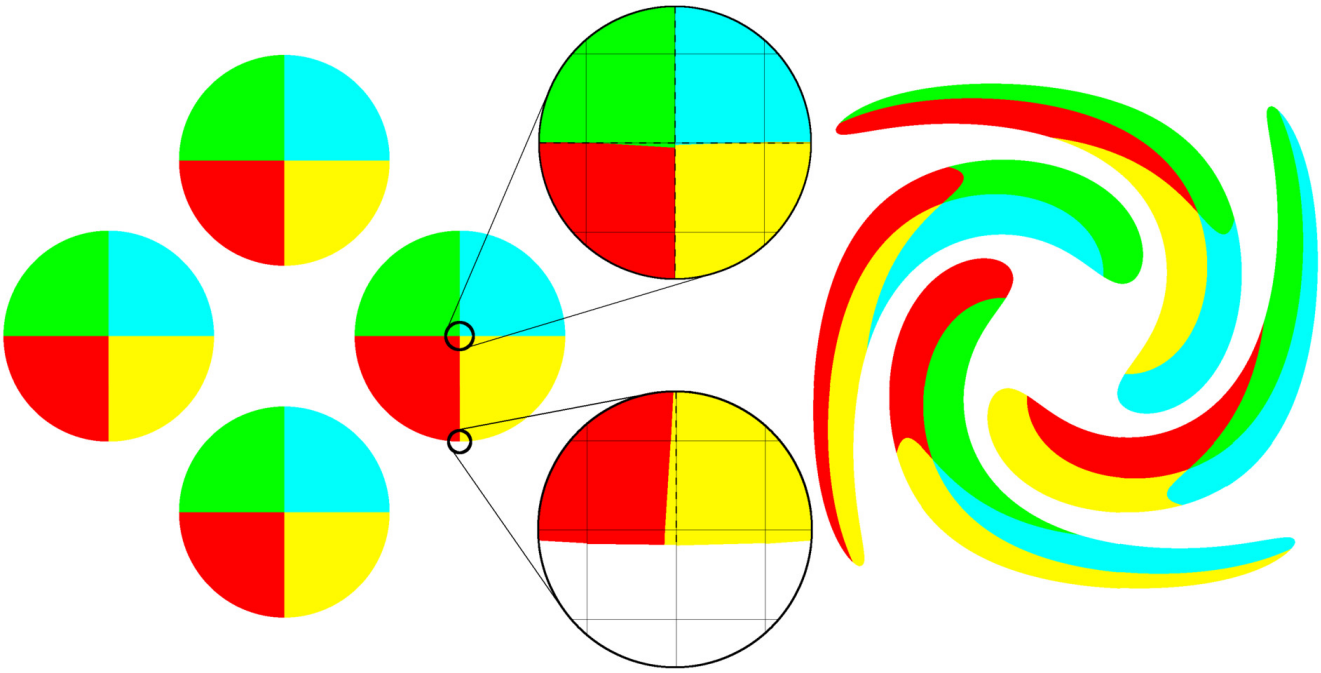


Fig. 6. MoF reconstruction of the dynamic test case on a 128^2 Cartesian grid. The left picture represents the initial and the final state and the right picture represents the maximal deformation. Magnified zones emphasize the defects at the final state compared to the initial state represented by dashed lines.

Table 2
Time ratio minimization/{analytic & minimization} for the dynamic test case on a 128^2 Cartesian grid.

	Min. 10^{-15}	Min. 10^{-12}	Min. 10^{-9}	Min. 10^{-6}
Ana. & Min. 10^{-15}	2.27	1.95	1.59	1.33
Ana. & Min. 10^{-12}	2.38	2.04	1.67	1.39
Ana. & Min. 10^{-9}	2.52	2.16	1.77	1.47
Ana. & Min. 10^{-6}	2.67	2.29	1.87	1.56

a tolerance value that gives an equivalent precision for the centroid defect, we observe that the analytic reconstruction is about 160% faster. With a tolerance value that would be used for a physical simulation (about 10^{-6}), we observe that the analytic reconstruction is at least 44% faster, but the centroid defect does not reach the machine error.

5.2. Dynamic reconstruction

In this numerical test we consider four disks, all composed of four materials, immersed in a reversible sheared flow. The domain is the unit square $[0, 1]^2$, the radius of the circles is 0.15 and they are arranged in a cross form with a distance of 0.25 from the center of the domain as represented in Fig. 6. The velocity field is the following:

$$\boldsymbol{u}(x, y, t) = \begin{bmatrix} -2\sin^2(\pi x)\sin(\pi y)\cos(\pi y) \\ 2\sin^2(\pi y)\sin(\pi x)\cos(\pi x) \end{bmatrix} \cos\left(\pi \frac{t}{2}\right) \tag{17}$$

The mesh is a Cartesian grid composed of 128^2 cells. The total number of iteration is 4,000 and the time step is 5×10^{-4}. We use the Lagrangian remap algorithm with a Runge–Kutta 2 scheme to advect the fluids as presented in [1].

In this numerical test, more than two fluids can be present in one cell. To reconstruct the interface, we use the serial dissection [4] where the analytic reconstruction is used for the first fluid and the minimization algorithm is used for the remaining fluids. When four fluids are involved simultaneously in a cell, we use the B-tree dissection [4] to generate more combinations. As a consequence, the minimization algorithm is always involved in the reconstruction. To compare the minimization to the analytic reconstruction, we compare the time to reconstruct the interface with only the minimization algorithm for various tolerance values to the time to reconstruct the interface with both algorithms involved and for various tolerance values.

Table 2 presents the time ratio between minimization and analytic & minimization reconstruction for tolerance values of 10^{-15}, 10^{-12}, 10^{-9} and 10^{-6}. Note that we only measure the time of reconstruction.

We observe that the time ratio is always greater than 1 which means that the analytic reconstruction combined to the minimization is always faster that minimization alone. In the worst case, when we compare the analytic reconstruction combined to the minimization with a tolerance value of 10^{-15} to the minimization alone with a tolerance value of 10^{-6}, we

observe that the proposed method is 33% faster than the classic minimization with a better precision. If we want to have the same precision with the classic minimization (10^{-15}), the time ratio becomes 2.67 for the benefit of the proposed method.

6. Conclusion

We have proposed an analytic MoF algorithm to reconstruct the interfaces on rectangular cells for two materials without minimization process. We have measured the CPU time of both the proposed algorithm and the minimization method on two test cases. The results show that the analytic reconstruction provides accurate results with a lower computational cost than the minimization method. When more than two materials are involved, the proposed algorithm can be applied only to the reconstruction of the first material, the remaining materials are reconstructed using the original minimization algorithm. Even in this case, we have shown that this algorithm is interesting in conjunction with the minimization algorithm.

Acknowledgements

This study was carried out with financial support from the French National Research Agency (ANR) in the framework of the "Investments for the future" Programme IdEx Bordeaux (ANR-10-IDEX-03-02), Cluster of excellence CPU.

The authors wish to thank the Aquitaine Regional Council for the financial support towards a 432-processor cluster investment, located in the I2M laboratory. Computer time for this study was also provided by the computing facilities MCIA (Mesocentre de Calcul Intensif Aquitain) of the Université de Bordeaux and of the Université de Pau et des Pays de l'Adour.

Appendix A. Flood algorithm for convex cells

Consider a convex polygon P composed of N vertices defined in counterclockwise order $\{\boldsymbol{p}_0, \cdots, \boldsymbol{p}_N\}$. The Flood Algorithm consists in finding a line segment Γ parametrized by its normal $\boldsymbol{n}$ and its distance to the origin $\xi^\star$ such that the part of the polygon behind the line segment matches a prescribed volume V, the flooding direction being given. The Algorithm 1 presents the Flood Algorithm we have implemented. The Fig. 7 illustrates the algorithm.

Algorithm 1: (Flood Algorithm) compute the distance $\xi^\star$ from the volume of fluid V and the flood direction $\boldsymbol{n}$.

input : Convex polygon $P = \{\boldsymbol{p}_0, \cdots, \boldsymbol{p}_N\}$, volume of fluid V, flood direction $\boldsymbol{n}$
output: Distance $\xi^\star$

Find the point of the polygon with the minimal signed distance in direction $\boldsymbol{n}$ and call it $\boldsymbol{p}_0$.

$l \leftarrow 1$, $r \leftarrow N$;
$|\Gamma| \leftarrow 0$, $|\Gamma_{\text{next}}| \leftarrow 0$;
$\xi \leftarrow 0$, $\xi_{\text{next}} \leftarrow 0$;
$V_{\text{trapezoid}} \leftarrow 0$;
for $k \leftarrow 1$ **to** $N - 1$ **do**
  if $\xi_r < \xi_l$ **then**
    $\xi_{\text{next}} \leftarrow \xi_r$;
    $r \leftarrow r + 1$;
  else
    $\xi_{\text{next}} \leftarrow \xi_l$;
    $l \leftarrow l - 1$;
  end
  $|\Gamma_{\text{next}}| \leftarrow$ `ComputeSection(`$\xi_{\text{next}}, \boldsymbol{n}, P$`)`;
  $V_{\text{trapezoid}} \leftarrow 0.5(\xi_{\text{next}} - \xi)(|\Gamma_{\text{next}}| + |\Gamma|)$;
  if $V_{\text{tot}} + V_{\text{trapezoid}} \geq V$ **then**
    $\alpha \leftarrow \dfrac{V - V_{\text{tot}}}{V_{\text{trapezoid}}}$;
    $\beta \leftarrow \sqrt{\left(\dfrac{|\Gamma|}{|\Gamma_{\text{next}}| + |\Gamma|}\right)^2 + \alpha \dfrac{|\Gamma_{\text{next}}| - |\Gamma|}{|\Gamma_{\text{next}}| + |\Gamma|}} + \dfrac{|\Gamma|}{|\Gamma_{\text{next}}| + |\Gamma|}$;
    $\xi^\star \leftarrow \xi + (\xi_{\text{next}} - \xi)\dfrac{\alpha}{\beta}$;
    break;
  end
  $V_{\text{tot}} \leftarrow V_{\text{tot}} + V_{\text{trapezoid}}$;
  $\Gamma \leftarrow \Gamma_{\text{next}}$;
  $\xi \leftarrow \xi_{\text{next}}$;
end

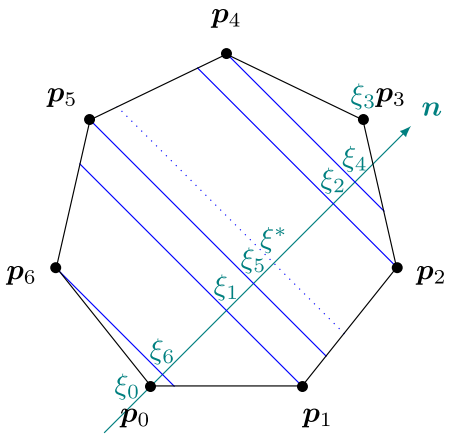


Fig. 7. Illustration of the Flood Algorithm on a convex polygon.

References

[1] V. Dyadechko, M. Shashkov, Moment-of-fluid interface reconstruction, Los Alamos National Laboratory Report LA-UR-05-7571.
[2] V. Dyadechko, M. Shashkov, Moment-of-fluid interface reconstruction, Los Alamos National Laboratory Report LA-UR-07-1537.
[3] H.T. Ahn, M. Shashkov, Multi-material interface reconstruction on generalized polyhedral meshes, J. Comput. Phys. 226 (2) (2007) 2096–2132.
[4] V. Dyadechko, M. Shashkov, Reconstruction of multi-material interfaces from moment data, J. Comput. Phys. 227 (11) (2008) 5361–5384.
[5] H.T. Ahn, M. Shashkov, Adaptive moment-of-fluid method, J. Comput. Phys. 228 (8) (2009) 2792–2821.
[6] S. Galera, J. Breil, P.-H. Maire, A 2d unstructured multi-material cell-centered arbitrary lagrangian–eulerian (ccale) scheme using mof interface reconstruction, Comput. Fluids 46 (1) (2011) 237–244.
[7] S.P. Schofield, M.A. Christon, Effects of element order and interface reconstruction in fem/volume-of-fluid incompressible flow simulation, Int. J. Numer. Methods Fluids 68 (11) (2012) 1422–1437.
[8] J. Breil, T. Harribey, P.-H. Maire, M. Shashkov, A multi-material reale method with mof interface reconstruction, Comput. Fluids 83 (2013) 115–125.
[9] M. Jemison, E. Loch, M. Sussman, M. Shashkov, M. Arienti, M. Ohta, Y. Wang, A coupled level set-moment of fluid method for incompressible two-phase flows, J. Sci. Comput. 54 (2–3) (2013) 454–491.
[10] R.N. Hill, M. Shashkov, The symmetric moment-of-fluid interface reconstruction algorithm, J. Comput. Phys. 249 (2013) 180–184.
[11] M. Jemison, M. Sussman, M. Arienti, Compressible, multiphase semi-implicit method with moment of fluid interface representation, J. Comput. Phys. 279 (2014) 182–217.
[12] M. Jemison, M. Sussman, M. Shashkov, Filament capturing with the multimaterial moment-of-fluid method, J. Comput. Phys. 285 (2015) 149–172.
[13] D.L. Youngs, Time-dependent multi-material flow with large fluid distortion, Numer. Methods Fluid Dyn. 24 (1982) 273–285.
[14] J.E. Pilliod, E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, J. Comput. Phys. 199 (2) (2004) 465–502.
[15] H. Anbarlooei, K. Mazaheri, 'Moment of fluid' interface reconstruction method in axisymmetric coordinates, Int. J. Numer. Methods Biomed. Eng. 27 (10) (2011) 1640–1651.
[16] M.B. Friess, J. Breil, S. Galera, P.-H. Maire, M. Shashkov, A multi-material ccale-mof approach in cylindrical geometry, arXiv preprint, arXiv:1111.4900.
[17] P. Strobach, Solving cubics by polynomial fitting, J. Comput. Appl. Math. 235 (9) (2011) 3033–3052.
[18] P. Strobach, The fast quartic solver, J. Comput. Appl. Math. 234 (10) (2010) 3007–3024.
[19] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[20] J. Breil, S. Galera, P.-H. Maire, A two-dimensional vof interface reconstruction in a multi-material cell-centered ale scheme, Int. J. Numer. Methods Fluids 65 (11–12) (2011) 1351–1364.