11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
E. Oñate, J. Oliver and A. Huerta (Eds)

A PARALLEL SECOND-ORDER CUT-CELL METHOD : VALIDATION AND SIMULATION AT MODERATE REYNOLDS NUMBERS

F. BOUCHON*, T. DUBOIS* and N. JAMES†

* Laboratoire de Mathématiques UMR 6620, Université Blaise Pascal,
Campus des Cézeaux - B.P. 80026, 63171 Aubière cedex, France,
Firstname.Name@math.univ-bpclermont.fr

† Laboratoire de Mathématiques et Applications UMR 7348, Université de Poitiers,
Téléport 2 - B.P. 30179, Boulevard Marie et Pierre Curie,
86962 Futuroscope Chasseneuil Cedex, France,
Nicolas.James@math.univ-poitiers.fr

Key words: Immersed boundary, cut-cell method, incompressible flow, parallel computing

Abstract. We present a parallel version of a second-order cut-cell scheme for the numerical simulation of two-dimensional incompressible flows past obstacles. The cut-cell method is based on the MAC scheme on cartesian grids and the solid boundary is embedded in the computational mesh. Discretizations of the viscous and convective terms are formulated in the context of finite volume methods ensuring local conservation properties of the scheme. Classical second-order centered schemes are applied in mesh cells which are *sufficiently* far from the obstacle. In the mesh cells cut by the obstacle, first-order approximations are used. While the scheme is locally first-order, that in the cut-cells, a global second-order accuracy is recovered. The time discretization is achieved with a second-order projection scheme. Due to the presence of solid boundaries, the linear systems for the velocity components and the pressure are non-symmetric but the stencil remains compact as in the classical MAC scheme on cartesian grids. They are solved by a direct method based on the capacitance matrix method and discrete Fourier transforms (DFT) in the direction tranverse to the mean flow. The parallel code is based on the MPI library for the communications between processes. The computational grid is splitted in the x-direction so that the DFTs are local to the MPI processes. A *divide and conquer* approach is applied to solve the tridiagonal systems whose solutions correspond to datas distributed accross all the MPI processes. The efficiency and robustness of the method are supported by numerical simulations of 2D flows past a circular cylinder at Reynolds number $\mathrm{Re} = 9\,500$ on grids with up to 212 millions of points ran on 48 MPI processes. Good agreement with published numerical results are obtained.

1 INTRODUCTION

To improve the discretization in the near-cylinder region, a new cut-cell method that ensures conservation properties (mass and kinetic energy) and directly enforces no-slip Dirichlet boundary conditions on an immobile obstacle was proposed in [1]. We focus here on a parallel version based on MPI which has been recently developped and its application to the numerical simulation of the flow past a circular cylinder at Reynolds number $\mathrm{Re} = 9\,500$.

The staggered arrangement of the velocity components is adapted to the cut-cell geometry (triangle, trapezoid, rectangle, pentagon). However, the pressure node is placed at the center of the cartesian cells for both fluid-cells and cut-cells. These locations of the unknowns induce specific discretizations in mesh cells located in the neighborhood of the solid boundary. As a consequence, the linear systems are non-symmetric.

The parallel version is based on a splitting of the datas among processes along the horizontal axis, namely each process works with a vertical band of the computational domain. As usual in local discretization methods, the parallel computation of the explicit terms, such as the nonlinearity, requires very few communications. The only tricky part concerns the non-symmetric linear systems. They are efficiently solved by a direct method based on the capacitance matrix method (see [1] for details) : the presence of the obstacle is treated as a perturbation added to the standard five-point stencil scheme obtained when the computational domain is fully filled by a fluid. Schematically, the algorithm consists in three steps. First, a linear system of dimension the number of cut-cells, which is much smaller than the total number of unknowns, is solved on one dedicated process of the MPI communicator. The involved matrix has been factorized during a preprocessing step performed once for all before time iterations. Then, a penta-diagonal linear system (similar to the system obtained without any solid obstacles) is solved. As each MPI process contains datas located on vertical bands, discrete Fourier transforms can be applied in the y-direction resulting in a collection of independant tridiagonal systems, for which the right-hand sides are distributed among MPI processes, due to the choice of the grid partioning. A parallel direct solver based on the *divide and conquer* (DAC) approach has been implemented. The DAC method is efficient here as we have to solve many tridiagonal systems, once per grid point in the y-direction, simultaneously. Sequential tasks, inherent to the DAC approach, are distributed among all the MPI processes avoiding useless waiting time. Finally a linear correction is applied in order to account for the presence of the obstacle in the computational domain. The parallel code reaches a good level of performance : less than 15% (in the worst cases) of the CPU time is spent in communications between processes. The sequential part performed on one process represents a negligible amount of CPU time.

The parallel cut-cell algorithm is applied to the numerical simulation of the flow past a circular cylinder at $\mathrm{Re} = 9\,500$ on grids with up to 212 millions of points and with up to 48 MPI processes. Good agreement with published numerical results are observed. The

computations presented here have been performed on a DELL cluster using up to 48 cores of Xeon processors. The nodes of the cluster are connected with a low latency bandwith network.

2 THE CUT-CELL METHOD

2.1 Preliminaries

We consider a two-dimensional domain $\Omega = (-L_x, L_x) \times (-L_y, L_y)$ in which a solid obstacle $\Omega^S \subset \Omega$ is enclosed. Our aim is to study the time evolution of an incompressible flow in $\Omega^F = \Omega \backslash \Omega^S$ and we assume that the motion of the solid boundary $\Gamma^S = \partial\Omega^S$ is prescribed. Therefore, the velocity field $\mathbf{u}(\mathbf{x}, t) = (u, v)$ at location $\mathbf{x} = (x, y) \in \Omega_F$ and time $t > 0$ is governed by the incompressible Navier-Stokes equations

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + \nabla(\mathbf{u} \otimes \mathbf{u}) + \nabla p = 0, \\ &\nabla \cdot \mathbf{u} = 0, \qquad \mathbf{u}(\mathbf{x}, t = 0) = \mathbf{u}_0, \end{aligned} \tag{1}$$

where $\mathbf{u}_0$ is an initial condition and $\nu > 0$ is the kinematic viscosity. Equations (1) are supplemented with boundary conditions on $\Gamma = \partial\Omega$. Dirichlet boundary conditions are imposed on the immersed boundary Γ^S.

The temporal discretization of (1) is achieved by using a second-order projection scheme. In a first step, momentum equations are advanced in time with a semi-implicit scheme decoupling the velocity and pressure unknowns. We use a second-order backward difference (BDF2) scheme for the time derivative and the viscous term combined with a second-order Adams-Bashforth scheme for the nonlinear terms. Then, the intermediate velocity is projected in order to obtain a free-divergence velocity field. Projection methods are efficient and are widely used for the numerical simulations of incompressible turbulent flows. The main feature of these schemes is that they allow to enforce the incompressibility constraint up to the computer accuracy : the discrete divergence is directly related to the residual of the pressure linear system.

2.2 Staggered grids

The computational domain Ω is discretized by a cartesian mesh with $n_x + 1$ (resp. $n_y + 1$) points in the horizontal (resp. vertical) direction. The mesh size h_y in the vertical direction is constant while a non-uniform mesh, generated by a regular mapping function, is used in the horizontal direction. The horizontal mesh size h_x equals h_y in a region containing the obstacle Ω^S. A discrete boundary Γ_h^S, which is piecewise linear on each computational cell, is used to approximate the immersed boundary Γ^S.

As in the classical MAC scheme for cartesian grids (see [2]), the discrete velocity components are located at the midpoints of the cell edges while the discrete scalar pressure is placed at the center of the cell. Let us consider a computational cell K_{ij}, we can distinguish two cases :

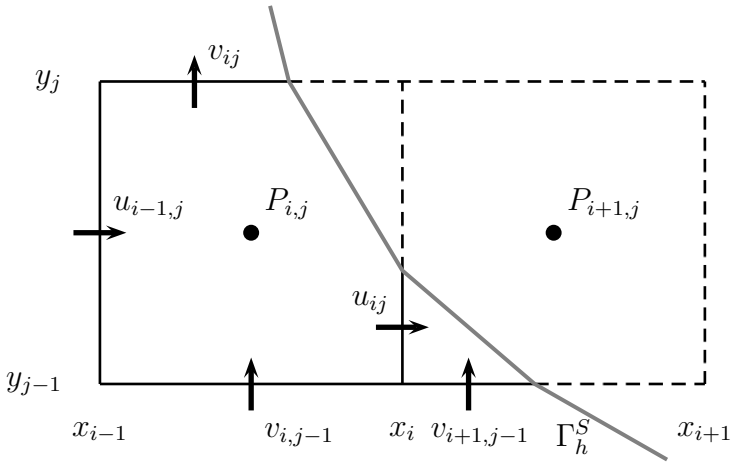

Figure 1: Location of the unknowns in the cut-cells K_{ij} (left) and $K_{i+1,j}$ (right). The dashed parts of the cell edges are located inside the obstacle.

1. K_{ij} is a fluid cell that is $K_{ij} \subset \Omega^F$: the staggered arrangement of the MAC scheme is applied ;
2. K_{ij} is a cut-cell that is $K_{ij} \cap \Omega^S \neq \emptyset$: the velocity unknowms are placed at the center of the part of the edges lying in the fluid zone while the pressure is left at the center of K_{ij} and can therefore be located inside the solid depending on how K_{ij} is geometrically splitted in fluid and solid parts.

Figure 1 summarizes this staggered arrangement of the discrete unknowns.

2.3 The cut-cell approximation of the spatial operators

On cartesian grids, the discretization of the viscous terms (as well as the nonlinear ones) by means of finite volume or finite difference approximations leads to symmetric discrete operators based on five-point stencils. With a staggered arrangement of the unknowns these stencils are specific for each unknown, that is the velocity components and the pressure. As the discrete solid boundary Γ_h^S intersects the computational mesh, the location of the discrete velocity components in cut-cells are changed compared with the classical MAC grid. Therefore, when one (or more) of these cut-cell unknowns belongs to the five-point stencil of a discrete operator, the numerical scheme has to be modified accordingly. Also, boundary conditions at the obstacle surface have to be accounted for in the numerical scheme.

The spatial discretization of the Navier-Stokes equations based on the staggered arrangement of the unknowns described above is fully detailed in [1]. Let us just mention that the stencil of the discrete viscous terms is enlarged to a six-point one for computational meshes close to Γ_h^S and first-order formulae are used. This locally first-order truncation error in the vicinity of the obstacle does not affect the global second-order convergence rate of the scheme.

The coefficients of the discrete operators depend on the geometry of the computational meshes, namely on the length of their edges. In cut-cells, these coefficients depend on the location of the intersection of Γ_h^S with the cartesian grid. As a consequence, the linear systems obtained to compute the velocity and the pressure are no longer symmetric. The efficiency, in terms of floating point operations, of the linear solver is essential in order to be able to simulate flows at large Reynolds numbers requiring a large number of mesh points.

2.4 A fast direct solver

We use a fast direct solver derived from the capacitance matrix method and adapted for the case of non uniform grids (see [1] and [3] for the details). We summarize the resulting algorithm. After spatial discretization of the Navier-Stokes equations, one linear equation is obtained per node in the part of the computational domain filled by the fluid and per unknown that is u, v and p. We complete these sets of linear equations by adding similar ones for nodes of the cartesian grid lying inside the solid obstacle but with zero as right-hand side. The unknowns corresponding to mesh points in Ω_h^S are fictitious ones. As in Ω^F, the numerical scheme accounts for the boundary conditions on Γ_h^S, the fluid unknowns are independant to the solid ones. We therefore obtain linear algebraic systems defined on the whole cartesian grid with size $(n_x - 1) \times n_y$ for u, $n_x \times (n_y - 1)$ for v and $(n_x - 1) \times (n_y - 1)$ for p. All three linear systems are similar in nature : the resulting matrices have similar structures with five or six non-zero coefficients per row.

Let us denote by $A \in \mathcal{M}_N(\mathbb{R})$ one of these matrices. Then at each time iteration, we have to solve a linear system

$$Ax \;=\; z \tag{2}$$

with z computed from the velocity and the pressure at previous time steps. As it is mentioned above, the matrix A is non-symmetric. Let us consider now the matrix G obtained with the same discretization on the whole computational Ω totally filled by a fluid that is with no obstacles. The matrices A and G differ only on rows corresponding to computational meshes for which the five-point stencil interacts with a cut-cell. Let us denote by n_c this total number of rows, namely rows such that $A - G$ have non-vanishing coefficients. The efficiency of our direct solver is due to the fact that n_c is small compared with N and that the non-zero coefficients on each row of $A - G$ is bounded. The linear system (2) can be rewritten as

$$Gx \;=\; z \;-\; Qy \tag{3}$$

where Q is a matrix of dimensions $N \times n_c$ with one non-vanishing coefficient per column, equal to one, and $y \in \mathbb{R}^{n_c}$ such that

$$Q\,y \;=\; (A - G)\,x.$$

It can be easily shown that y is solution of the following linear system

$$\left(I_{n_c} \;+\; M\,G^{-1}Q\right)\; y \;=\; M\,G^{-1}\,z \tag{4}$$

with $M = Q^t(A - G)$. The matrix $I_{n_c} + M\,G^{-1}Q$ is a non-singular matrix (see [3] for a proof) of size n_c.

Based on these relations, the algorithm implemented to solve (2) consists in a preprocessing step where the matrix $I_{n_c} + M\,G^{-1}Q$ is factorized (we use a LU-factorization) followed by

i) Compute z and solve $Gw = z$;

ii) Compute Mw and solve (4) ;

iii) Compute Qy and solve $Gx = z - Qy$.

Recalling that G is the matrix corresponding to the standard MAC scheme on the whole computational mesh, steps i) and iii) can be performed by using any efficient solvers available on cartesian grids. In the present work, we use Discrete Fourier transforms in the vertical direction (where the mesh is uniform) combined with LU-factorizations of the resulting tridiagonal systems.

2.5 A parallel version of the linear solver

The parallel version of this cut-cell method is based on explicit communications performed by calling functions of the MPI library. The main feature of MPI is that a parallel application consists in running p independant processes which may be executed on different computers, processors or cores. These processes can exchange datas by sending/receiving messages *via* a network connecting all the involved computing units.

The first step when developping a parallel algorithm is to define a suitable and efficient splitting of the datas among the MPI processes : each MPI process will treat datas associated with a part of the total computational mesh. For our problem, this choice is straightforward and is related to the algorithm used to solve the linear systems. Indeed, it is much easier to implement a parallel resolution of tridiagonal linear systems rather than a parallel version of the DFT. Therefore, the parallel version of the code is based on a splitting of the datas along the horizontal axis, so that each MPI process works with a vertical slice of the computational mesh as it is illustrated on Figure 2.

In the framework of finite volume or finite difference schemes on cartesian grids, the explicit computation of spatial derivatives is local and involves very few communications. The only tricky part concerns the resolution of the linear systems. The step ii) of the direct solver described in the previous section consists in solving a linear system involving the matrix $I_{n_c} + M\,G^{-1}Q$. As the LU-factorization of this matrix has been computed and stored in a pre-processing step at the beginning of the time iterations, we have to solve two triangular systems which can not be efficiently performed on parallel computers. As n_c is small compared to the size of the global problem, we choose to dedicate this task to one given MPI process, fixed in advance. Once the linear system is solved, the resulting vector is scattered from this MPI process to the other processes.

As it was mentioned in the previous section, linear systems of steps i) and ii) are solved by first applying a DFT in the y-direction : these computations are independant and can be performed without any communications due to the distribution of datas among the MPI processes. This results in a collection, one per grid point in the y-direction, of independant tridiagonal linear systems connecting all nodes of the mesh in the x-direction. A parallel direct solver based on the *divide and conquer approach* (DAC) has been implemented. The DAC method, applied to solve one tridiagonal linear system on $n_p > 1$ MPI processes, consists in splitting the tridiagonal matrix into n_p independant blocks (one per MPI process). The solutions of these systems have to be corrected in order to recover the solution of the global system. These corrections correspond to $2n_p - 1$ values which are solutions of a tridiagonal linear system of size $2n_p - 1$. This phase of the DAC method is sequential and has to be performed on one process inducing a useless waiting time for the other processes. However, as we have to solve n_y such systems simultaneously, this sequential part can be distributed among all the n_p processes. In this context, the DAC algorithm leads to an efficient parallel code.

The parallel code has a good level of performance : less than 15% of the CPU time is spent in communications between MPI processes. The sequential part performed on one process represents a negligible amount of CPU time. The computations presented here have been performed on a DELL cluster using up to 48 cores of Xeon processors. A low latency bandwith network connects the cluster nodes.

3 NUMERICAL RESULTS

3.1 Assessment of the cut-cell scheme

Recently, a validation tool was proposed in [4] in order to investigate the numerical errors associated with immersed boundary methods. The assessment procedure is based on a reference solution (see [5]) for the classical problem of the flow past a circular cylinder at Reynolds number $\mathrm{Re} = 40$. The grid convergence for the velocity and pressure obtained with our cut-cell method is illustrated in Figure 3 : a second-order convergence rate is found for both quantities.

3.2 Flow past a circular cylinder at $Re = 9500$

Preliminary numerical results for the flow past a circular cylinder at Reynolds number $\mathrm{Re} = 9500$ were obtained with the sequential code and were presented in [1]. A graphical comparison with the pictures of the experiments performed by Bouard and Coutenceau, published in [6], revealed a good qualitative agreement. Further investigations demonstrate that the computational domain $\Omega = (-5, 5) \times (-2.5, 2.5)$ used in [1] was too small : a discripency of the drag coefficient with the results of the simulations performed by Koumoutsakos and Leonard [7] was observed. In our simulation, the drag coefficient in the early stage of the flow development was over-estimated by a factor of the order of 15%.

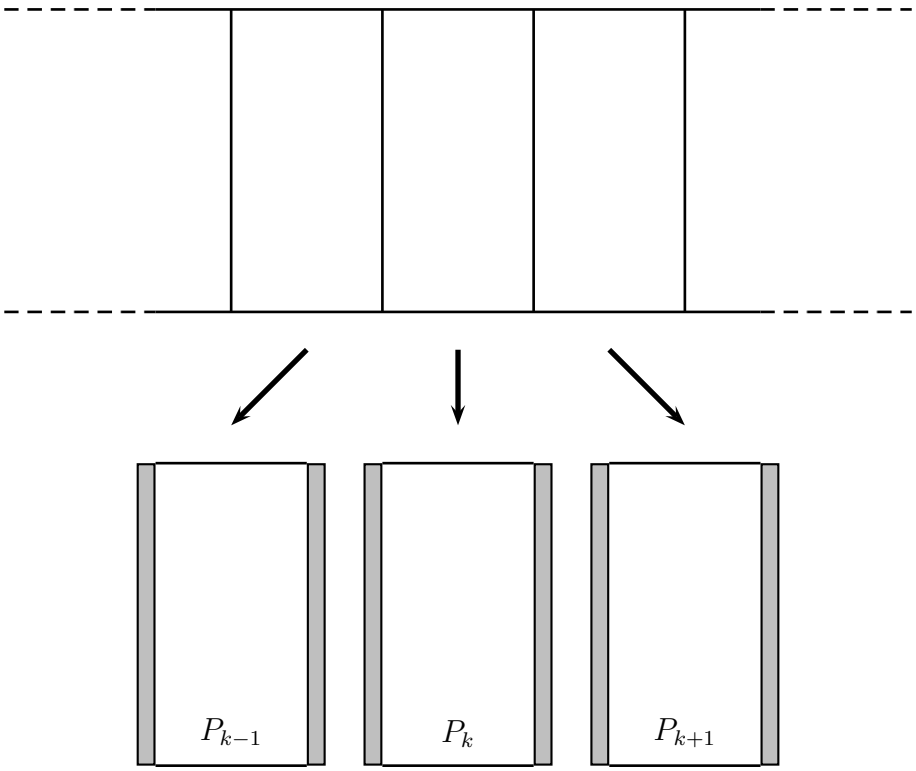

Figure 2: Splitting of the computational grid among the MPI processes P_k, $k = 0, \ldots, n_p - 1$. The lightgray aera indicates extra additional storage required for communications between neighborhood processes.

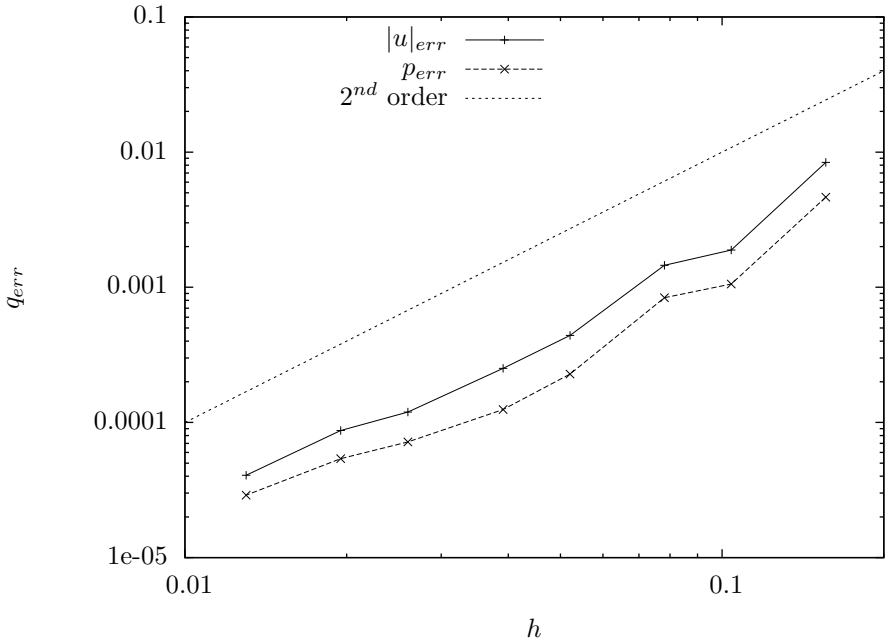

Figure 3: Velocity and pressure errors in L^2-norm as functions of the spatial resolution.

In order to be able to increase the size of Ω, and therefore the total number of mesh points and so the amount of computations, parallel simulations were necessary. We have recently performed several numerical simulations listed in Table 1 using 32 and 48 MPI processes. The elapsed time (in second) reported in Table 1 is of course meant per time iteration. The number of cut-cells n_c reported in Table 1 corresponds the velocity grids. For the pressure, n_c is slightly smaller. Note that the number of cut-cells n_c is much smaller than the total number of points $n_x \times n_y$. The elapsed time corresponding to the preprocessing step of the direct solver, namely the computation and assembling of the matrix $(I_{n_c} + MG^{-1}Q)$ in (4), is equal to 24 minutes per unknown, that is u, v and p, for the finest simulation F_{10}. On Figure 4, the time evolution of the drag coefficient for our simulations as well as the Koumoutsakos and Leonard's (KL's) [7] datas are plotted. We note that results of the C_{10} and C_{20} simulations are almost identical indicating that a convergence with respect to the size of the computational domain has been reached at this resolution, namely $h_y = 2.5 \times 10^{-3}$. A good agreement is found with the KL's results up to $t \simeq 1.5$. For larger values of the non-dimensional time, the order of magnitude of the drag coefficient for these two simulations are correct but the time evolution differs slightly than the KL's results.

The simulation F_{10} performed on a two times finer mesh, that is $h = 1.25 \times 10^{-3}$, follows with a good accuracy the KL's datas up to $t = 3.0$. In order to reach a grid convergence of the simulations, further investigations are in progress and will presented at the conference.

Simulation	L	n_x	n_y	h_y	δt	n_c	n_p	Elapsed time (s)
C_{10}	10	4096	8192	0.0025	0.00025	1980	32	4.73
C_{20}	20	8640	16384	0.0025	0.00025	1980	32	6.77
F_{10}	10	12960	16384	0.0012	0.0002	3960	48	7.52

Table 1: Numerical parameters of the parallel simulations of flow past a cylinder at Re = 9500. The computational domain is $\Omega = (-L, L)^2$, the cylinder diameter is equal to 1.0 and n_p is the number of MPI processes.

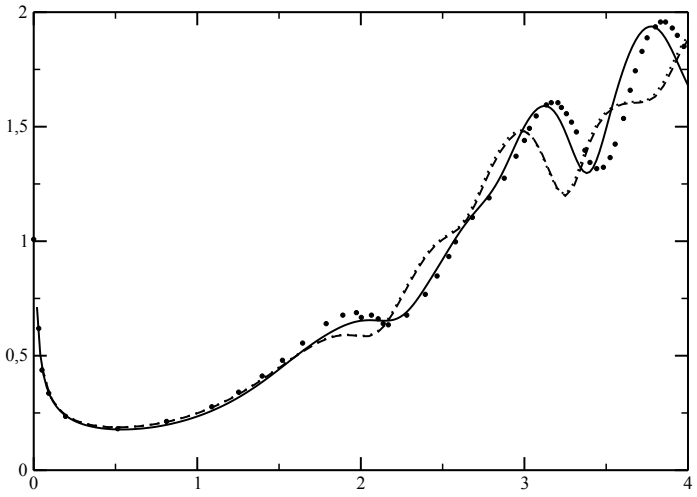

Figure 4: Time evolution of the drag coefficient obtained for the simulations C_{10} (dotted line), C_{20} (dashed line) and F_{10} (solid line) compared with the numerical results of Koumoutsakos and Leonard [7] (black dots).

4 Acknowledgments

The numerical simulations presented in this paper were performed on the DELL cluster (24 nodes with 2 XEON processors linked with an InfiniBand network) of the Laboratoire de Mathématiques de l'Université Blaise Pascal.

REFERENCES

[1] F. Bouchon, T. Dubois and N. James, A second-order cut-cell method for the numerical simulation of 2D flows past obstacles, *Computers & Fluids*, 80–91, 2012.

[2] F.H. Harlow and J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Phys. Fluids*, **12**(8), 2182–2189, 1965.

[3] F. Bouchon and G. H. Peichl, The Immersed Interface Technique for Parabolic Problems with Mixed Boundary Conditions, *SIAM J. Num. Anal.*, **48**, 2247–2266, 2010.

[4] N. James , D. Biau, J. Dambrine, M. Pierre and E. Lamballais, Assessment of Immersed Boundary Method toward high accuracy, *EUROMECH / ERCOFTAC Colloquium 549*, Leiden, The Netherlands, 2013.

[5] R. Gautier, D. Biau, and E. Lamballais. A reference solution of the flow over a circular cylinder at Re = 40, *Computers & Fluids*, **75**, 103–111, 2013.

[6] R. Bouard and M. Coutenceau, The early stages of development of the wake behind an impulsively started cylinder for $40 < Re < 10^4$, *J. Fluid. Mech.*, **101**, 583–607, 1980.

[7] P. Koumoutsakos and A. Leonard, High Resolution Simulations of the Flow Around and Impulsively Started Cylinder Using Vortex Methods, *J. Fluid Mech.*, **296**, 1–38, 1995.