An efficient iterative penalization method using recycled Krylov subspaces and its application to impulsively started flows

T. Gillis[a,*], G. Winckelmans[a], P. Chatelain[a]

[a] *Institute of Mechanics, Materials and Civil Engineering, Université catholique de Louvain 1348 Louvain-la-Neuve, Belgium*

Abstract

We formulate the penalization problem inside a vortex particle-mesh method as a linear system. This system has to be solved at every wall boundary condition enforcement within a time step. Furthermore, because the underlying problem is a Poisson problem, the solution of this linear system is computationally expensive. For its solution, we here use a recycling iterative solver, rBiCGStab, in order to reduce the number of iterations and therefore decrease the computational cost of the penalization step. For the recycled subspace, we use the orthonormalized previous solutions as only the right hand side changes from the solution at one time to the next. This method is validated against benchmark results: the impulsively started cylinder, with validation at low Reynolds number (Re = 550) and computational savings assessments at moderate Reynolds number (Re = 9500); then on a flat plate benchmark (Re = 1000). By improving the convergence behavior, the approach greatly reduces the computational cost of iterative penalization, at a moderate cost in memory overhead.

Keywords: Brinkman penalization, Krylov subspaces recycling, Recycling BiCGStab, Vortex methods, Particle-mesh methods, Linear solver

1. Introduction

Vortex methods use the vorticity-velocity formulation of the (incompressible) Navier-Stokes equations. Re-meshed vortex particle (VP) methods have been shown to efficiently and accurately simulate complex viscous vortex flows [1–5]. The Lagrangian treatment of the convection term, combined with the re-meshing operation leads to a method with accurate treatment of the convection. The re-meshed and hybrid approach, as in [2, 3, 6–10], here called the Vortex Particle-Mesh (VPM) method, sometimes also called the Vortex-In-Cell (VIC) method, combines the advantage of a particle method with those of a mesh-based approach. Several techniques have been proposed for embedding solid bodies with arbitrary geometries in VPM methods [4, 5]. Most recent efforts use Immersed Interface method (IIM) [11, 12] or penalization techniques [13–15]. For the latter, an iterative penalization method has been proposed by Hejlesen et al. [16], in order to improve the accuracy at the boundary, avoid diffusion errors [17] and keep the time-step sufficiently large. However, this iterative procedure is highly expensive since, at each iteration, one needs to recover velocity from vorticity by solving a Poisson equation. When considering unbounded problems, this can be carried out on a small sub-domain around the body; yet it still remains the bottleneck operation of such method. Moreover, in a parallel computing context, this domain change implies some important data

*Corresponding author
Email address: thomas.gillis@uclouvain.be (Thomas Gillis)

mappings in order to maintain acceptable load balancing within the penalization sub-problem. In this work, we propose to formulate the penalization problem as a linear system and to use advanced iterative methods to reduce the number of iterations needed by the penalization sub-step.

Iterative methods for generic linear systems as $Ax = b$ are widespread. When the matrix A is Hermitian, one of the prominent algorithms, the conjugate gradient method (CG) [18], uses short 3-term recurrences to build the solution iteratively. This implies that the storage overhead is low enough to be used in CFD applications. To solve more general systems, one usually uses the GMRes algorithm [18, 19]. Since this method is based on the construction of an orthogonal basis of the Krylov subspace at each iteration, the cost in memory and orthogonalization increases with the number of iterations. Therefore, another approach has also been proposed in the literature: transform short recurrences from the CG algorithm to make it suitable for arbitrary matrices. Two generalization of CG based on bi-orthogonalization were proposed: BiCG and QMR. BiCG, however, requires two matrix - vector products involving A and A^T. To overcome this limitation, Sonneveld [20] proposed a transpose-free variant: the conjugate gradient squared algorithm (CGS) which improves the convergence by using the "wasted" matrix A^T - vector multiplication. Yet, this variation can exhibit a build-up of rounding errors and erratic convergence. An improved version has then been proposed by van der Vorst [21]: BiCGStab, which stabilizes the convergence behavior. This method rapidly became famous due to its low computational cost and simplicity.

The integration of time-dependent PDEs very often involves a linear system solution at each time step; this can be due to either an elliptic sub-problem (as in the present fluid mechanics problem) or the use of implicit time integration. Linear solvers are thus ubiquitous in computational science and constitute a focal point of research in applied mathematics. When using an iterative solver, the computational cost will be directly governed by the number of iterations needed to converge. In the context of a time-dependent problem, this motivates the recycling of the Krylov subspaces obtained during the previous solution. Generalized conjugate residual method with inner orthogonalization (GCRO) [22], GMRes and BiCGStab with deflated restarting [23–26] and GCRO with optimal truncation (GCROT) [27] are as many examples of such algorithms. Assuming a continuous behavior of the PDE and its solution in time, Amritkar et al. [28] proposed a recycling solver based on the combination of both GMRes with deflated restarting and recycling BiCGStab and applied it to standard CFD problems such as turbulent channel flows and flows through porous media. We further extend this latter work to the penalization method in the vorticity formulation of the incompressible Navier-Stokes equations. The computational gains are assessed on the salient and enduring challenge of taking complex geometries into account with VPM methods in an efficient and flexible manner.

In this paper, we present the transposition of the recycling BiCGStab to the penalization in VPM methods to reduce the number of iterations needed in order to converge. We assess its computational gains on benchmark problems: the number of calls to the Poisson solver, the convergence and the memory overhead. We first briefly recall the Vortex Particle-Mesh (VPM) method and the relevant algorithmic techniques, then we detail the proposed penalization algorithm in Section 2. Sections 3, 4.1 and 4.2 present an assessment of the computational efficiency of our method. We close this paper in Section 5 with a conclusion and perspectives for future work.

2. Methodology

2.1. Vortex Particle-Mesh (VPM) method and Brinkman penalization (2D)

We briefly present Vortex Particle-Mesh methods, the use of penalization techniques and the way the penalization step is solved using iterative methods. Particle-mesh methods are well suited for solving advection dominated problems. The VPM method is a prime exemple; it solves the vorticity formulation of the incompressible Navier-Stokes equations [2, 3, 9, 10, 16]. In 2D, we only have the out-of-plane vorticity, $\boldsymbol{\omega} = \omega \mathbf{e}_z$, and

$$\frac{\mathrm{D}\omega}{\mathrm{D}t} = \frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \, \nabla^2 \omega \quad , \tag{1}$$

where $\mathbf{u}$ is the velocity which has to be recovered through the Poisson problem,

$$\nabla^2 \mathbf{u} = -\nabla \times \boldsymbol{\omega} \quad . \tag{2}$$

The vorticity field, ω, is discretized by means of particles with positions $\mathbf{x}_p$, material surfaces S_p and integral of vorticity $\Gamma_p = \int_{S_p} \omega \; d\mathbf{x} \triangleq \omega_p \; S_p$. The mesh is used to carry out operations such as the solution of (2) and operator computations [2, 9], while the time integration is carried on the particles. The VPM method then solves the following set of equations

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x}_p \\ \omega_p \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \nu \, \nabla^2 \omega \end{bmatrix}_p \quad . \tag{3}$$

As a consequence, particle-mesh methods need to interpolate the vorticity field and the right-hand sides of (3) between the particles and the mesh.

Brinkman penalization methods [14, 16] embed the object in the fluid domain and enforce the velocity inside the obstacle, Ω_b, to be $\mathbf{u} = \mathbf{u}_b$, where $\mathbf{u}_b$ is the desired velocity inside the immersed body. It adds this constraint to the Navier-Stokes problem using an additional penalization term, which will behave as a relaxation of any spurious velocity inside the object. Since the constraint is imposed on the velocity, these methods are naturally applied to pressure-velocity formulation of the Navier-Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \nu \, \nabla^2 \mathbf{u} + \lambda \, \chi \, (\mathbf{u}_b - \mathbf{u}) \quad , \tag{4}$$

where λ is the Lagrange multiplier and χ is the mask function taken as being

$$\chi \, (\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_b, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Applying the curl operator to Eq. (4) leads to the vorticity formulation. In 2D, one then obtains the penalized form of the Navier-Stokes equations [29]:

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \, \nabla^2 \omega + \lambda \, \nabla \times [\chi \, (\mathbf{u}_b - \mathbf{u})] \quad . \tag{6}$$

For each particle, we then solve the following set of equations

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}_p\\ \omega_p\end{bmatrix} = \begin{bmatrix}\mathbf{u}\\ \nu\,\nabla^2\omega + \lambda\,\nabla\times\left[\chi\left(\mathbf{u}_b-\mathbf{u}\right)\right]\end{bmatrix}_p \quad . \tag{7}$$

These ordinary time-differential equations are typically time-integrated using a second or higher order scheme (eventually using the first order Euler scheme for the diffusion part) and the velocity field is recovered by solving Eq. (2). Solving this Poisson problem is the bottleneck of such methods. For the unbounded domain considered in this paper, this is handled by using the Hockney-Eastwood algorithm which works on the Fourier transform of the zero-padded doubled-domain source field [30, 31].

The time integration of Eq. (7) is carried out through a splitting approach [32]. Given $\mathbf{u}^n$, ω^n and Δt, one goes through the following substeps:

1. Convection of the particles and interpolation of the vorticity onto the mesh (or remeshing).
2. Diffusion on the mesh (here presented using the Euler scheme)

$$\tilde{\omega}^{n+1} = \omega^n + \Delta t\ \left[\nu\,\nabla^2\omega^n\right] \quad . \tag{8}$$

3. Computation of the unpenalized velocity field, on the mesh, by solving

$$\nabla^2\,\tilde{\mathbf{u}}^{n+1} = -\nabla\times\tilde{\boldsymbol{\omega}}^{n+1} \quad . \tag{9}$$

4. Computation of the correction vorticity, on the mesh

$$\delta\omega = \mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) \quad , \tag{10}$$

   where $\mathcal{F}$ is a function that varies from one penalization technique to another.
5. Update of the vorticity

$$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega \tag{11}$$

   and interpolate on the particles.

The function $\mathcal{F}$ effectively produces the vortex sheet, $\delta\omega$, that cancels the velocity field inside the body. This leads to the observation that penalization is very similar to the problem solved using boundary elements methods (BEM) [1, 4, 5], as such BEM method also computes the vortex sheet required at each time step in order to cancel the slip velocity at the interface [5]. Both methods aim to capture the generation of the same vortex sheet but through different formulations and discretizations (explicitly, with panels or on a grid). A first widespread approach [15, 29, 32, 33] uses the following function

$$\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right) = \nabla\times\left[\frac{\tilde{\mathbf{u}}^{n+1} + \lambda\Delta t\ \ \chi\ \mathbf{u}_b}{1+\lambda\Delta t\ \ \chi}\right] - \tilde{\omega}^{n+1} \quad , \tag{12}$$

which comes from an implicit Euler integration of the penalization sub-problem

$$\frac{\partial u}{\partial t} = \lambda\chi\left(\mathbf{u}_b - \mathbf{u}\right) \quad . \tag{13}$$

Because the implicit Euler scheme is unconditionally stable, one can use an arbitrary high value for the Lagrange multiplier. Nevertheless, this method has some major drawbacks:

1. It generates substantial diffusion errors, as claimed by Rasmussen et al. [17];
2. As explained by Hejlesen et al. [16], it does not provide a global relation between vorticity and velocity which leads to a dramatically decreasing time step in order to provide accurate results.

Since getting the correct vortex sheet is crucial for capturing unsteady behaviors such as the added mass effects of accelerated objects, Hejlesen et al. [16] proposed another way to evaluate $\mathcal{F}\left(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}\right)$. The authors suggest to solve the following problem: at time step n, given a non-penalized vorticity field, $\tilde{\omega}^{n+1}$, (and an associated non-penalized velocity field, $\tilde{\mathbf{u}}^{n+1}$); the iterative penalization method aims at finding the vorticity correction, $\delta\omega$, which generates the velocity correction, $\delta\mathbf{u}^{n+1}$, such that $\tilde{\mathbf{u}}^{n+1} + \delta\mathbf{u} = \mathbf{u}^{n+1} = \mathbf{u}_b$ in Ω_b. The method proposed by Hejlesen et al. [16] iterates on this vorticity correction to converge to consistent vorticity and velocity fields (see algorithm 1). We formulate this problem as a linear system

$$A\left(\delta\boldsymbol{\omega}\right) = \nabla \times \left[\chi\left(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1}\right)\right] \quad , \tag{14}$$

where $\delta\boldsymbol{\omega} = \delta\omega\mathbf{e}_z$ and the linear operator A is composed as

$$A : \delta\omega_k \rightarrow \nabla \times \left[\chi\delta\mathbf{u}_k\right] = \nabla \times \left[\chi\left(\left(\nabla^2\right)^{-1}\left(-\nabla \times \delta\boldsymbol{\omega}_k\right)\right)\right] \quad . \tag{15}$$

We notice that this operator is non-symmetric because of the χ function. If we pursue the above analogy with the corresponding boundary integral equation in two dimensions, we expect the operator A to have a non-void kernel. The uniqueness of the solution is therefore not ensured. Indeed, the solution of Eq. (14) is defined up to a constant which sets the circulation around the object. Nevertheless, as reported by various authors [16, 17, 29, 32, 33] and observed here, one does not need to impose the vorticity integral explicitly.

Within this framework, the iterative method proposed by Hejlesen et al. [16] can be reinterpreted as an over-relaxed Jacobi iteration. The stability constraint reported by the authors is in fact the well known stability constraint for this kind of method

$$0 < \alpha = \lambda\Delta t < 2 \quad . \tag{16}$$

Crucially, a single matrix-vector product, $A\ \delta\omega_k$, is expensive: therein lies the solution of a Poisson equation, as shown in Eq. (15). Because each iteration of the penalization involves such an expensive component, this potentially makes the boundary condition enforcement the bottleneck of the global algorithm. To reduce the cost, Hejlesen et al. [16] proposed to reduce the computational domain of the penalization method. When using a FFT-based Poisson solver, this domain restriction can only be performed in the unbounded direction. Still, it reduces the computational and memory overheads due to the penalization. Nevertheless, a crucial issue remains: a Jacobi

Algorithm 1: Penalization algorithm - Jacobi(α) from Hejlesen et al. [16]

Given: $\tilde{\mathbf{u}}^{n+1}$, $\tilde{\omega}^{n+1}$,
Set: $\delta\omega_0 = 0$; $\delta\mathbf{u}_0 = 0$
Compute: $b = \nabla \times \left[\chi\left(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1}\right)\right]$; $r_0 = b$

for $\underline{k = 1, 2, ...}$ **do**

$$\delta\omega_k = (1-\alpha)\,\delta\omega_{k-1} + \alpha\, r_{k-1}$$
$$\nabla^2\left(\delta\mathbf{u}_k\right) = -\nabla \times \delta\boldsymbol{\omega}_k$$
$$r_k = \nabla \times \left[\chi\left(\mathbf{u}_b - \left(\tilde{\mathbf{u}}^{n+1} + \delta\mathbf{u}_k\right)\right)\right]$$

end

$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega$

method will require more iterations compared to more advanced algorithms, making its cost significant with respect to the rest of the time-step. One needs then to reduce the number of iterations needed to reach convergence and hence, there is a strong motivation to reduce the number of iterations.

2.2. Krylov based iterative methods

In this section, we briefly present the BiCGStab algorithm and the way to recycle Krylov subspaces with this method. We refer to [18] for the fundamentals on this class of methods.

We consider a non-Hermitian matrix, $A \in \mathbb{R}^{n\times n}$. One of the most famous algorithm for such matrices is GMRes. It consists in computing iteratively an orthogonal basis, V, of $\mathcal{K}(A)$ such that, at each iteration,

$$V^T A V = H \qquad \text{where} \qquad V^T V = I \tag{17}$$

and where H is a Hessenberg matrix (i.e $H_{ij} = 0$ if $i > j+1$) . Alternatively, there is the BiCGStab method. It is a member of the bi-conjugated algorithm family which replaces the Hessenberg matrix, H, by a tridiagonal one using bi-orthogonalization. In other words, we bi-orthogonalize A

$$W^T A V = T \qquad \text{where} \qquad \begin{cases} W^T V = I \\ A V = V\, T \\ A^T W = W\, T^T \end{cases}, \tag{18}$$

where T is the tridiagonal matrix. One of the first bi-orthogonalized variants of the Conjugated Gradients (CG) method has been proposed by Lanczos [34]: Bi-Conjugated Gradients (BCG). This method iteratively computes a solution of the system $A\,x = b$ using 4 vectors r_k, r_k^*, p_k and p_k^*, defined as

$$\begin{aligned} r_k &= \phi_k(A)\, r_0 & r_k^* &= \phi_k\left(A^T\right) r_0^* \\ p_k &= \pi_k(A)\, r_0 & p_k^* &= \pi_k\left(A^T\right) r_0^* \end{aligned}, \tag{19}$$

where ϕ_k is a certain polynomial of degree k which satisfies $\phi_k(0) = 1$ and π_k is another polynomial of degree k.

The series of ϕ and π polynomials are initialized as $\phi_0 = \pi_0 = 1$. This algorithm then iteratively computes the solutions using short recurrences [18]

$$\begin{aligned} \alpha_k &= \frac{\left(\phi_k(A)\, r_0\ ,\ \phi_k\left(A^T\right) r_0^*\right)}{\left(A\pi_k(A)\, r_0\ ,\ \pi_k(A^T)\, r_0^*\right)} \\ \phi_{k+1}(x) &= \phi_k(x) - \alpha_k\ x\ \pi_k(x) \\ \beta_k &= \frac{\left(\phi_{k+1}(A)\, r_0\ ,\ \phi_{k+1}\left(A^T\right) r_0^*\right)}{\left(\phi_k(A)\, r_0\ ,\ \phi_k(A^T)\, r_0^*\right)} \\ \pi_{k+1}(x) &= \phi_{k+1}(x) + \beta_k\ \pi_k(x)\ . \end{aligned} \tag{20}$$

The drawback of this method is to use 2 matrix-vector products (one with A and one with A^T) at each iteration to compute α_k and β_k. To take advantage of this wasted matrix-vector product, Sonneveld [20] proposed the Conjugated Gradient Squared (CGS) algorithm which is based on the observation that α_k can be rewritten as

$$\alpha_k = \frac{\left(\phi_k^2(A)\, r_0\ ,\ r_0^*\right)}{\left(A\ \pi_k^2(A)\, r_0\ ,\ r_0^*\right)}\ . \tag{21}$$

Therefore it seeks for a residual defined by

$$r_k = \phi_k^2(A)\, r_0\ . \tag{22}$$

However, this method shows poor convergence behavior. To stabilize CGS, van der Vorst [21] introduced BiCGStab, a stabilized version where the residual is defined as

$$r_k = \psi_k(A)\, \phi_k(A)\, r_0\ , \tag{23}$$

where $\psi_k(A)$ is a polynomial assembled iteratively and chosen to smooth the convergence. This new ψ polynomial avoids the use of the squared ϕ, which damages the convergence behavior. It endows the method with an improved convergence behavior but, its convergence is not guaranteed to be monotonic anymore.

2.3. Recycling methods

The recycling of information from one solution attempt to the next, on a perturbed system, has become a much pursued area of research in linear algebra. All recycling methods use the same framework: they first compute the best solution within a known space and then solve the remaining part of the system [24, 26, 28]. In order to do so, they rely on 2 subspaces: $\mathcal{U}$, and $\mathcal{C}$, both of dimension l, related by $A : \mathcal{U} \to \mathcal{C}$. Recycling methods store 2 bases which span those 2 subspaces: $U \in \mathbb{R}^{n\times l}$ and $C \in \mathbb{R}^{n\times l}$, respectively. Given those two bases, an initial solution, x_{-1} and its associated residual, $r_{-1} = b - Ax_{-1}$, recycling methods first look for the best initial guess, $x_0 = x_{-1} + Uy$, where $y \in \mathbb{R}^l$, that verifies

$$\mathbf{x}_0 = \arg\min_y \left\|r_0\right\|_2\ , \tag{24}$$

subject to the constraints

$$AU = C \qquad \text{and} \qquad C^T C = I\ . \tag{25}$$

This problem becomes

$$y_0 = \arg\min_y \left\| r_{-1} - AUy \right\|_2 = \arg\min_y \left\| r_{-1} - Cy \right\|_2 = C^T r_{-1} \ . \tag{26}$$

The optimal solution, x_0, and its residual are then given by the following expressions [23]

$$x_0 = x_{-1} + UC^T r_{-1} \qquad r_0 = r_{-1} - CC^T r_{-1} \ . \tag{27}$$

The original problem $Ax = b$ has then become

$$A\left(x_0 + \zeta\right) = b \qquad \Rightarrow \qquad A\zeta = r_0 \ , \tag{28}$$

which we still have to solve. This is done using widespread algorithms like BiCGStab. However, since x_0 is the best solution in $\mathcal{U}$, we know that $r_0 \in \mathcal{C}^\perp$. To enforce this on Eq. (28) and inside the iterative solver, we orthogonalize the remaining problem with respect to $\mathcal{C}$ and solve the modified problem

$$A_C \overline{\zeta} = \left[I - CC^T\right] A\overline{\zeta} = \overline{r_0} = r_0 \ . \tag{29}$$

Since we solved Eq. (29) instead of Eq. (28) we have to recover ζ from $\overline{\zeta}$, which is done using

$$x = x_0 + \zeta = x_0 + \left[I - UC^T A\right] \overline{\zeta} \ , \tag{30}$$

as presented in [23].

2.4. Application of recycling methods to penalization in the VPM method

Instead of using a Jacobi iteration to solve the penalization part of Eq. (6), we propose to use the rBiCGStab algorithm of Section 2.3, that recycles solutions from previous time steps. We observe that the solution of the penalization problem, Eq. (14), does not change much from one time step to another and therefore previous solutions are accurate initial guesses. This rBiCGStab(l) algorithm will then store up to l previous solutions and use them to compute the best initial iterate possible (algorithm 2). What might seem like an expensive memory overhead is actually limited as the support of the corrective vorticity consists in the body and its direct surroundings. Additionally, for an unbounded flow, we can also reduce the computational domain, as done in [16]. Algorithm 2 has two major advantages. First, the memory cost is constant with the number of iterations, which is not the case for a GMRes-like method. Second, performing the small orthogonalization is cheaper than solving a Poisson equation ($\mathcal{O}\left(l\ n\right)$ versus $\mathcal{O}\left(n\ \log n\right)$ where l is chosen such that $l << n$). Furthermore, in this version of the algorithm, the orthogonalization is carried out using a modified Gram-Schmidt algorithm.

We note some drawbacks, however. First, it uses two matrix vector products per iteration which makes it more expensive than the Jacobi method per iterate. Still, the number of iterations needed to reach convergence will be much lower. Second, the residual does not converge monotically as with a GMRes-like method. This is a well-known behavior of this method and it is due to the use of bi-orthogonalization [18].

Algorithm 2: Penalization algorithm - rBiCGStab(l)

Given: $\tilde{\mathbf{u}}^{n+1}$, $\tilde{\omega}^{n+1}$, $U \in \mathbb{R}^{n\times l}$, $C \in \mathbb{R}^{n\times l}$ such that $AU = C$ and $C^TC = I$

Set: $k = 0$; $\rho_{old} = \rho = 0$

Compute: $b = \nabla \times \left[\chi\left(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1}\right)\right]$; $\quad \xi_0 = -\left[C^T b\right]$; $\quad r_0 = b - C\xi$; $\quad \beta = \left\|r_0\right\|$

Choose: $\hat{r} = r_0$

for $k = 1, 2, ...$ **do**

$\rho = \hat{r}^T r_k$

if $\rho == 0$ **then** Breakdown occured. **exit**

if $i == 0$ **then**

$p = r_k$

else

$\beta = \dfrac{\rho}{\rho_{old}}\dfrac{\alpha}{\omega}$

$p = r_k + \beta\left(p - \omega\nu\right)$

end

$\nu = Ap$; $\quad \eta = C^T\nu$; $\quad \nu = \nu - C\nu$

$\alpha = \dfrac{\rho}{\hat{r}^T\nu}$

$\xi_{k+1} = \xi_k + \alpha\eta$; $\quad x_{k+1} = x_k + \alpha p$; $\quad r_{k+1} = r_k - \alpha\nu$

if $E_2^{k+1} \leq \text{tol}$ **then** **exit**; **else** $k = k + 1$

$t = Ar_k$; $\quad \eta = C^T t$; $\quad t = t - C\eta$

$\omega = \dfrac{t^T r_k}{t^T t}$

$\xi_{k+1} = \xi_k + \omega\eta$; $\quad x_{k+1} = x_k + \omega r_k$; $\quad r_{k+1} = r_k - \omega t$

$\rho_{old} = \rho$

if $E_2^{k+1} \leq \text{tol}$ **then** **exit**; **else** $k = k + 1$

end

$\delta\omega = x_{k_{end}} - U\xi_{k_{end}}$

if $\delta\omega \neq U_{1:l}$ **then**

$C_{1:l} = \left[C_{2:l} \quad b - r_{k_{end}}\right]$; $\quad U = \left[U_{2:l} \quad \delta\omega^{n+1}\right]$

$C_{1:l} = QR$

$U_{1:l} = UR^{-1}$ $\quad C_{1:l} = Q$

end

$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega$

2.5. Convergence criterion for iterative penalization method

Every iterative method needs a stopping criterion. We recall that the stopping criterion should be an estimator of the solution quality. In Hejlesen et al. [16] the authors propose a way to compute the error at each iteration as the correction vorticity field's first moment integral (i.e. the impulse). This stopping criterion is however not general as it allows spurious solutions, which are even-symmetric with respect to the inflow direction. For these reasons, we rather use a dimensionless penalization residual. At a given iteration, k, we compute it as follows

$$E_2^k\left(r_k\right) = \frac{1}{\omega_\infty}\sqrt{\frac{1}{S_{\Omega_b}}\left\|r_k\right\|_2^2}\ , \tag{31}$$

where $\omega_\infty = U_\infty / L_{\Omega_b}$, S_{Ω_b} is the obstacle reference surface, L_{Ω_b} is the obstacle reference length and the 2 norm is taken as $\left\|a\right\|_2^2 = \sum_{i,j} a_{i,j}^2 h^2$ and $r_k = \nabla \times \left[\chi \left(\mathbf{u}_b - \left(\tilde{\mathbf{u}}^{n+1} + \delta \mathbf{u}_k\right)\right)\right]$.

2.6. Force computation

To compute the force exerted by the fluid on the immersed body, we use the momentum formulation from Noca et al. [35], as proposed by Mimeau et al. [15]: considering a control volume, Ω_c, that contains the obstacle, we have that

$$\mathbf{F} = -\frac{d}{dt}\int_{\Omega_c} \mathbf{u}\, d\mathbf{x} + \int_{\partial\Omega_c} \boldsymbol{\gamma}\cdot\hat{\mathbf{n}}\, d\mathbf{x} \ , \tag{32}$$

where $\hat{\mathbf{n}}$ is normal to $S = \delta\Omega_c$ and

$$\begin{aligned}\boldsymbol{\gamma} = \frac{1}{2}\left|\mathbf{u}\right|^2 \mathbf{I} \quad & -\frac{1}{N-1}\mathbf{u}\left(\mathbf{x}\times\boldsymbol{\omega}\right) + \frac{1}{N-1}\boldsymbol{\omega}\left(\mathbf{x}\times\mathbf{u}\right) \\ & -\frac{1}{N-1}\left[\left(\mathbf{x}\cdot\frac{\partial\mathbf{u}}{\partial t}\right)\mathbf{I} - \mathbf{x}\frac{\partial\mathbf{u}}{\partial t}\right] \\ & +\frac{1}{N-1}\left[\left[\mathbf{x}\cdot\left(\nabla\cdot\mathbf{T}\right)\right]\mathbf{I} - \mathbf{x}\left(\nabla\cdot\mathbf{T}\right)\right] + \mathbf{T} \ .\end{aligned} \tag{33}$$

in which N is the spatial dimension (2 or 3), $\mathbf{I}$ is the unit tensor and $\mathbf{T}$ is the viscous stress tensor, given by $\mathbf{T} = \mu\left[\nabla\mathbf{u} + \left(\nabla\mathbf{u}\right)^T\right]$. In our case, Ω_c is a rectangular box containing the body, Ω_b.

3. Validation and analysis of the presented method

In this section, we use the flow past a cylinder at low Reynolds number (Re = 550) to validate our approach and analyze its behavior. We use an adaptative time step, well suited for the impulsively started flow considered here. The time step, Δt, is constrained by the following accuracy conditions:

$$\Delta t \left\|S_{ij}\right\|_1 \leq C_1 \qquad \text{and} \qquad \Delta t \left\|\omega\right\|_1 \leq C_2 \ , \tag{34}$$

where $S_{ij} = \frac{1}{2}\left[\nabla\mathbf{u} + \left(\nabla\mathbf{u}\right)^T\right]$ is the strain tensor and $C_1 \simeq C_2 \simeq 0.15 \ldots 0.25$ are typical values. Throughout this work, we consider a cylinder of diameter D in an computational domain of size $[5D \times 5D]$ and we use the following dimensionless variables:

$$\omega^* = \frac{\omega\, D}{U_\infty} \qquad t^* = \frac{tU_\infty}{D} \qquad \mathrm{Re} = \frac{DU_\infty}{\nu} \qquad C_D = \frac{\mathbf{F}\cdot\mathbf{e}_x}{\frac{1}{2}\rho U_\infty^2 D} \ . \tag{35}$$

Furthermore, we compute the penalization residual as being

$$E_2^k\left(r_k\right) = \frac{D}{U_\infty}\sqrt{\frac{4}{\pi D^2}\left\|r_k\right\|_2^2} \ , \tag{36}$$

*3.1. Validation on the flow past an impulsively started cylinder at low Reynolds number (*Re = 550*)*

For this validation, we consider that the iterative procedure reached convergence when $E_2^k < 10^{-4}$. To put this convergence criterion in perspective, one should keep in mind that, following the Falkner-Skan theory, the momentum thickness of the boundary layer near the stagnation point is given by $\theta/D \simeq 0.1462/\sqrt{\mathrm{Re}}$. Using

this thickness instead of D in the scaling coefficient $\frac{D}{U_\infty}$ would give a new convergence criterion of $< 6.23\ 10^{-7}$, which is more representative of the physical interpretation behind the residual, but it requires to have a priori information about the flow solution. The resolution of the full domain is $\frac{\Delta \mathbf{x}}{5D} = \frac{1}{1024}$ and the subdomain used for the penalization problem, $\left[\frac{5}{4}D\ ;\ \frac{5}{4}D\right]$, is centered around the cylinder. The time step coefficients are taken as $C_1 = C_2 = 0.25$. We compare our drag coefficient against three references in the literature: Koumoutsakos and Leonard [1] (KL), Ploumhans and Winckelmans [4] (PW) and Marichal et al.[11, 12] (M). An analytic solution valid for short times also exists [36] and is given by

$$C_D = 4\left(\frac{\pi}{\mathrm{Re}\ t^*}\right)^{1/2} + 2\pi\left(9 - \frac{15}{\sqrt{\pi}}\right)\frac{1}{\mathrm{Re}}\ . \tag{37}$$

Fig. 1 shows the evolution of the drag coefficient, defined at Eq. (35), for short times.

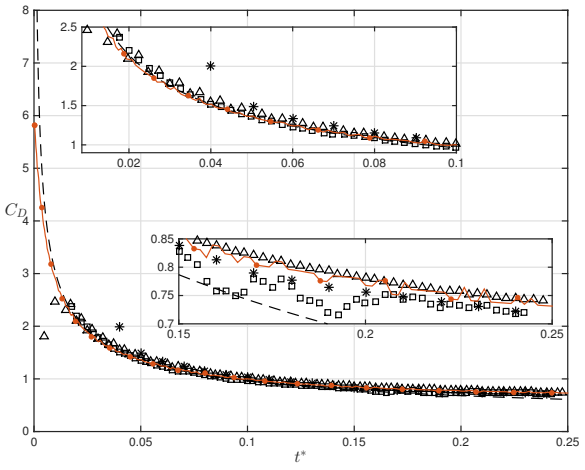

Figure 1: Impulsively started cylinder at Re = 550: C_D, penalization rBiCGStab(4) $(-\bullet-)$, KL [1] $(*)$, PW [4] $(\square)$, M [11, 12] and PhD thesis [37] $(\triangle)$ and Eq. (37) $(--)$. The first time step is $\Delta t^* = 10^{-6}$.

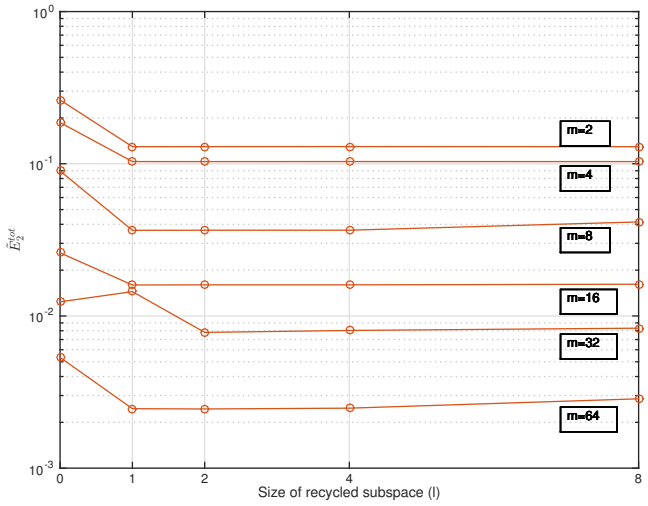

Figure 2: Impulsively started cylinder at Re = 550: $\tilde{E}_2^{tot}$ as a function of the memory overhead.

As expected, we underestimate C_D at the very early times (the theoretical behavior is singular at $t = 0^+$, corresponding to an infinitely thin boundary layer). Soon after, our simulation is well-resolved and the C_D curve falls back onto the theoretical curve, (momentarily though, as this curve is only valid for short times). Our results also compare favorably with all the references results of PW [4], KL [1] and M [11, 12]. The present method also shows a good agreement with the IIM Vortex Method of Marichal et al. [11], which performs the highest order treatment of the wall boundary condition among the reference results. It is important to note that our results are not filtered and still exhibits a very low noise signal, unlike those of PW [4] who used filtering (visible in the zoom $t^* \in [0.15\ ;\ 0.25]$).

*3.2. Analysis of the computational cost for the presented methods on the flow past an impulsively started cylinder at low Reynolds number (*Re = 550*)*

In this section, we compare the computational costs required by Jacobi(1.9) and rBiCGStab. The resolution of the full domain is $\frac{\Delta \mathbf{x}}{5D} = \frac{1}{256}$, the penalization subdomain and the time step coefficients are as in Section 3.1. The error allowed inside the body (and therefore the norm of the residual) will lead to a incorrect vorticity flux at the

wall and therefore will potentially pollute the flow solution outside the body. Following Eq. (31) and given a fixed time t^*, we compute the so-called "external error" as

$$\tilde{E}_2^{t^*}(m,l) = \frac{D}{U_\infty}\sqrt{\frac{4}{\pi D^2}\left\| \left(\omega^{t^*}(m,l) - \omega_{ref}^{t^*}\right)(1-\chi)\right\|_2^2}\ , \tag{38}$$

where m is the number of matrix-vector products, l is the dimension of the recycled subspace and $\omega_{ref}^{t^*}$ is the reference solution at the given time t^*. We chose $\omega_{ref}^{t^*}$ as the solution obtained by iterating till breakdown or convergence to a tolerance of 10^{-10}. Breakdown refers to a situation where the algorithm finds itself stuck: such a condition arises when the residual r is aligned with the conjugated residual $\hat{r}$. In the test performed for this case, breakdown solutions are still well converged ($E_2 \leq 10^{-8}$). However, this could be easily solved by restarting the algorithm and therefore picking another $\hat{r}$. Since we are interested only in starting times, we define the "short-time total external error" as being

$$\tilde{E}_2^{tot}(m,l) = \sum_{n=0}^{10} \tilde{E}_2^{(n\Delta t^*)}(m,l) \qquad \text{with} \quad \Delta t^* = 0.01\ . \tag{39}$$

3.2.1. Influence of the number of iterations performed on the flow accuracy

The CPU-cost is directly driven by the number of matrix-vector (matvec) products computed in the iterative process. In Fig. 3, we further analyze the relationship between the computational cost invested in the penalization step and the error level in the flow, as defined by Eq. (38). The convergence slopes for $\tilde{E}_2^{0^+}$, $\tilde{E}_2^{0.1}$ and $\tilde{E}_2^{tot}$ are shown in Table 1. There are several notable differences between the behaviors of Jacobi and rBiCGStab-based methods.

	Jacobi(α)	rBiCGStab(l)				
	$\alpha = 1.9$	$l = 0$	$l = 1$	$l = 2$	$l = 4$	l =8
$\tilde{E}_2^{0^+}$	-0.68	-1.16	-1.16	-1.16	-1.16	-1.16
$\tilde{E}_2^{0.1}$	-0.80	-1.21	-1.02	-1.14	-0.89	-0.41
$\tilde{E}_2^{tot}$	-0.71	-1.19	-1.09	-1.17	-1.17	-1.14

Table 1: Impulsively started cylinder at Re = 550: convergence slopes for different algorithms based on the $\tilde{E}_2^{t^*}$ error defined by Eq. (38) and the $\tilde{E}_2^{tot}$ error defined by Eq. (39).

First we observe that the slopes of Jacobi are $\simeq 0.7$, while the slopes for rBiCGStab are $\simeq 1.1$, highlighting a real computational gain for rBiCGStab (cfr Table 1). We also notice, on the left part of Fig. 3, that the recycled subspace is very useful since it allows to reduce the mean error in the flow with non-CPU overhead. Of course, storing additional solutions on the subdomain slightly increases the memory footprint, an overhead quite acceptable in view of the accuracy and CPU gains. Second, we observe, on the right part of Fig. 3, that reusing subspaces and using rBiCGStab can lead to non monotonic convergence behavior (which, we recall, is inherent to the bi-conjugated methods). However, the error is still lower that the error performed by Jacobi. As expected, we also see that the gain in accuracy is more important at the very beginning of the simulation than when the flow is more developed.

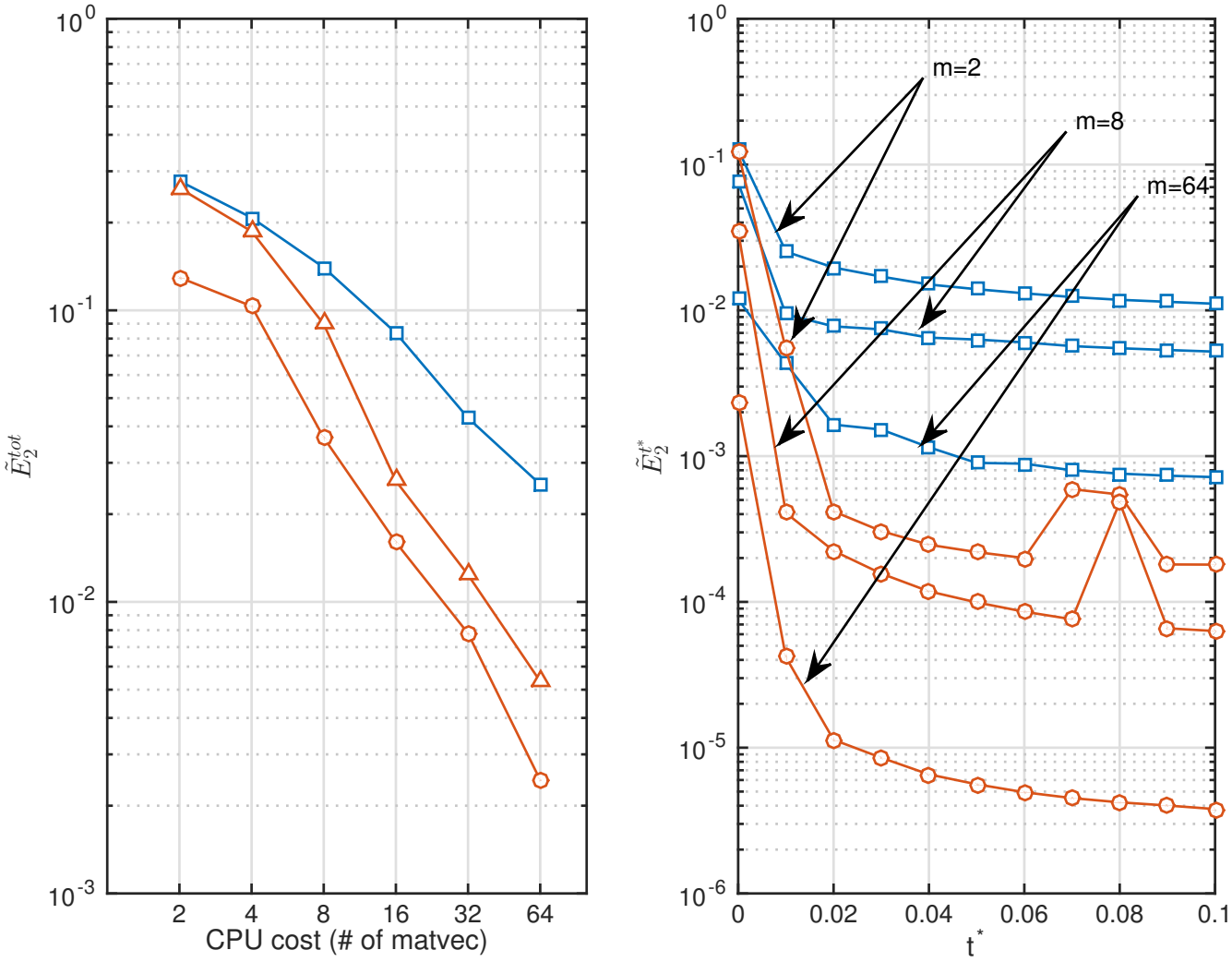

Figure 3: Impulsively started cylinder at Re = 550: (left) $\tilde{E}_2^{tot}$ as a function of the CPU cost (which is directly proportional to the number of matvecs). (Right) $\tilde{E}_2^{tot}$ as a function of time for Jacobi and rBiCGStab(2). Jacobi(1.9) ($-\square-$), rBiCGStab(0) ($-\triangle-$) and rBiCGStab(2) ($-\circ-$)

3.2.2. Influence of the memory overhead on the recycled solution

The next parameter of interest is the dimension of the subspace to recycle, as presented on Fig. 2. We see that the advantage of taking l previous solutions into account quickly decreases with l. In the present case, choosing a value of $l = 2$ puts us in the plateau area and guarantees that we are not involved in the non-monotonic behavior of BiCGStab. We conclude that, even when using a "large" time-step (i.e. $C_1 = C_2 = 0.25$), two previous solutions are enough for computational acceleration and do not imply a large memory overhead. However, this conclusion needs to be put in perspective since we consider very early times with a marked unsteadiness at $t = 0^+$. After this strong transient however, the flow development is smoother and therefore easily captured by the previous solution.

4. Results

*4.1. Impulsively started flow past a cylinder at high Reynolds number (*Re = 9500*)*

Vortex methods are well suited for moderately high Reynolds flows. In this section, we present results for the flow past a cylinder at Re = 9500 and assess the computational gain. We consider the algorithms Jacobi(1.9) and rBiCGStab(2). The resolution of the full domain is $\frac{\Delta \mathbf{x}}{5D} = \frac{1}{4096}$. The penalization subdomain stays the same as in Section 3.1 and the time step coefficients satisfies Eq. (34) with the same coefficients as in Section 3.1. We consider that both methods have to reach an error of $E_2 \leq 10^{-2}$, as defined by Eq. (36), which allows a relatively converged flow solution at reasonable CPU-cost.

Figure 4 compares the flow obtained for the two methods considered. The drag coefficients, C_D, shown on Fig. 5, both exhibit a non-smooth behavior, typical of the method we used to compute the drag force (see Section 2.6). One way to increase the smoothness consists in performing more iterations, thereby decreasing the residual error within the object. Furthermore, one notes that our results compare favorably with those of the references [1, 16].

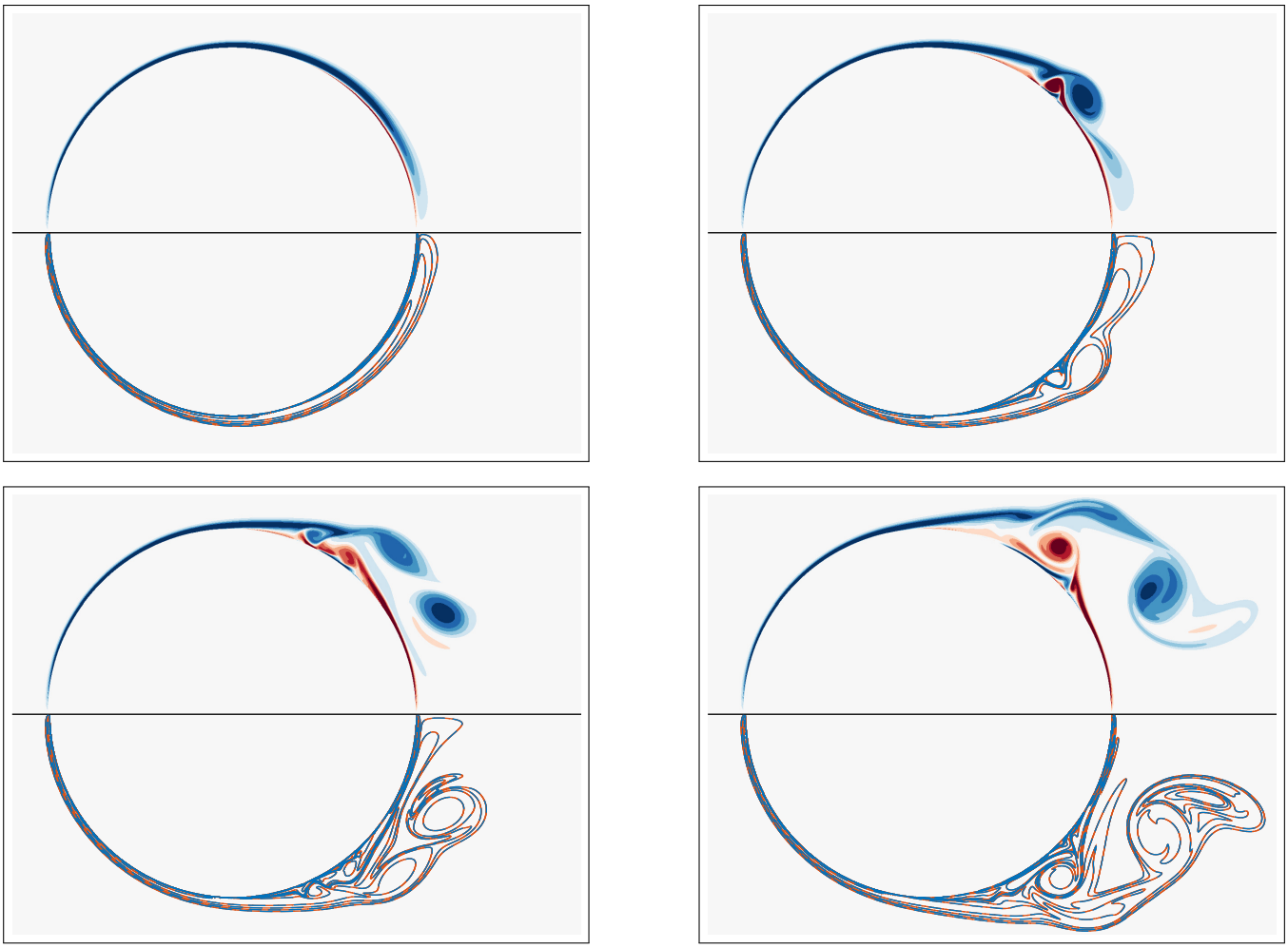

Figure 4: Impulsively started cylinder at Re = 9500 ($t^* = 0.5$, $t^* = 1.0$, $t^* = 1.5$, $t^* = 2.0$): vorticity, rBiCGStab(2) (up and down in orange) and Jacobi(1.9) (down in blue), both with tolerance 10^{-2}

Fig. 6 shows the number of matrix-vector products needed to converge the solution down to the chosen tolerance. We observe that using rBiCGStab(2) reduces the average number of iteration needed by a factor of $\simeq 48$ ($\simeq 22.6$ for Jacobi(1.9) and $\simeq 0.47$ for rBiCGStab(2)). Recycling solutions is therefore very effective at speeding up the computation, even for high Reynolds number flows.

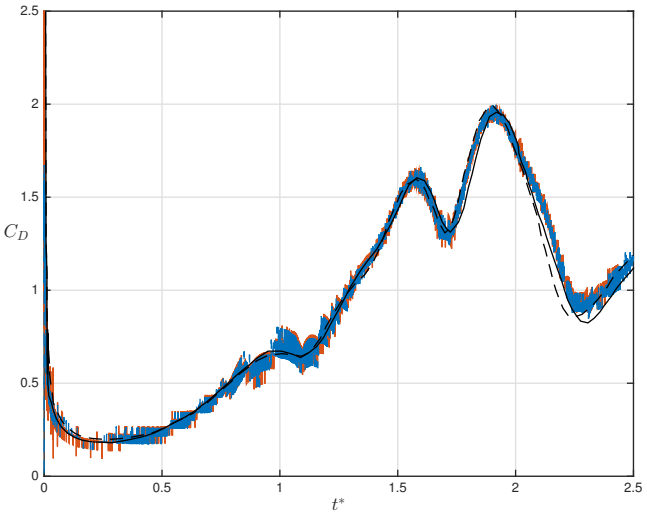

Figure 5: Impulsively started cylinder at Re = 9500: C_D, Jacobi(1.9) (blue ––), rBiCGStab(2) (orange –), Hejlesen et al. [16] (black ––) and Koumoutsakos and Leonard [1] (black –).

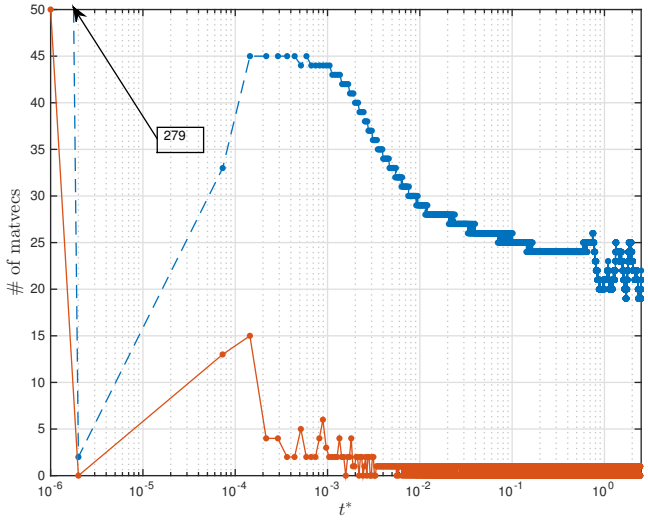

Figure 6: Impulsively started cylinder at Re = 9500: comparison of the number of matvec needed to converge to a tolerance of 0.01 for Jacobi(1.9) (––•–) and rBiCGStab(2) (–•–). The first time step of Jacobi(1.9) is needs 279 matvecs.

*4.2. Impulsively started flow past a thin plate at moderate Reynolds number (*Re = 1000*)*

In this section, we assess the computational gains on a challenging benchmark characterized by sharp corners and non-negligeable acceleration effects, due to impulsively start, that, typically, can pose problems to a penalization method. The resolution of the full domain is $\frac{2\Delta \mathbf{x}}{5L} = \frac{1}{2048}$, where L is the height of the plate; the penalization subdomain is $(32\Delta \mathbf{x} \times 1024\Delta \mathbf{x})$; the time step coefficients are taken as $C_1 = C_2 = 0.15$ and the flat plate thickness is $T = 8\Delta \mathbf{x}$, as in [16]. The flat plate constitutes is an stiff problem to solve, especially at early times. It is indeed challenging for the penalization method to respond to the very important temporal variations of this flow, when, furthermore, the penalization term support is a thin region: Hejlesen et al. [16] reported that more than 100 iterations are needed in order to converge. Although this convergence criterion is based on the drag coefficient (the iteration stops when the increment of the drag coefficient is less than 0.001 of the drag itself), one could argue that this is too computationally expensive to make it usable within a realistic framework. In our case, we define the error E_2^k, as

$$E_2^k\left(r_k\right) = \frac{L}{U_\infty}\sqrt{\frac{1}{LT}\left\|r_k\right\|_2^2} \ . \tag{40}$$

The tolerance on this error for both rBiCGStab(2) and Jacobi(1.9) is 10^{-2}. The Reynolds number is defined as $\mathrm{Re} = \dfrac{LU_\infty}{\nu} = 1000$.

The large scales of the flow are shown in Fig. 7 for both solvers, while Fig. 8 shows the evolution of the unfiltered drag coefficient, C_D.

At early times, the use of the rBiCGStab method leads to drag coefficient results that are a bit more spread than those obtained using the Jacobi method. Let us recall that this problem is very stiff and especially at $t = 0^+$. We observe that iterating through the penalization method (and therefore performing more matrix-vector products, like the Jacobi method does) helps to smooth the drag coefficient behavior. Furthermore, one notes that our results for both methods compare favorably with the reference [16], except at early times, where we observe a faster-decreasing drag than Hejlesen et al. [16].

Moreover, one must keep in mind that the drag coefficient is obtained through a control volume integration, which involves the first moment of vorticity. If the edges of the penalized vorticity, which actually are in the control volume, oscillate (which is very likely in the present very stiff problem), we expect to observe noise within the drag behavior at early times.

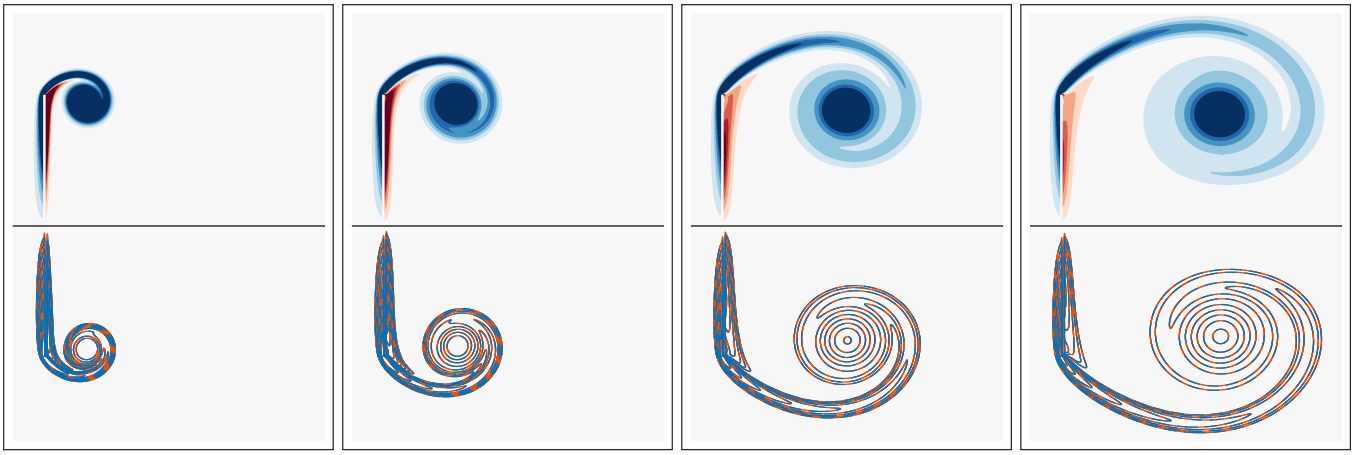

Figure 7: Impulsively started flat plate at Re = 1000 ($t^* = 0.25$, $t^* = 0.5$, $t^* = 1.0$, $t^* = 1.5$): vorticity, rBiCGStab(2) (up and down in orange) and Jacobi(1.9) (down in blue), both with tolerance 10^{-2}

Again, the computational costs are noticeably different between the two approaches: Jacobi(1.9) uses $\simeq 70$ matrix-vector products on average while rBiCGStab rapidly falls to an average of 0.55 matrix-vector products on average after the transient (tol $= 10^{-2}$). rBiCGStab only needs a relatively large number of iterations during the first few time steps, which cover the early times of the impulsive start. Yet, we note that this number remains one order of magnitude below that needed by Jacobi(1.9). The computational gain is thus significant and comes at the cost of a moderate memory overhead. It is indeed here quite limited due to the uses of a penalization-specific subdomain and of the small dimension of the recycled subspace ($l = 2$).

A closer inspection of the solutions inside the body reveals some differences between the two solvers. Fig. 10 shows the solutions in the vicinity of a plate tip. Both solvers exhibit a spurious vorticity within the plate thickness (which, recall, only covers 8 grid points). Yet this spurious vorticity is clearly more intense in the case of Jacobi(1.9); it also gets dampened more slowly.

In other tests (not shown here) performed with a larger time-step (i.e. $C_1 = C_2 = 0.25$), we observe oscillations in the drag coefficient results obtained using the rBiCGStab method. These oscillations indicate that the recycling method is then sensitive to the Lagrangian deformation of the particles which is large due to the corners of the plate, especially at early times. For such stiff test-cases, the time-step should therefore be wisely chosen, or a

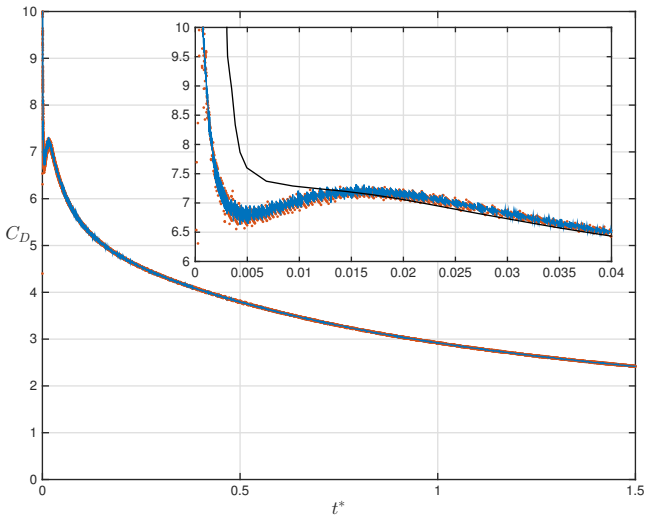

Figure 8: Impulsively started flat plate at Re = 1000: C_D, rBiCGStab(2) (orange •) and Jacobi(1.9) (blue −) both with tolerance 10^{-2}. For comparison purpose, we also represent, on the zoom, the drag curve from [16] ($T/L = 100$) (black −).

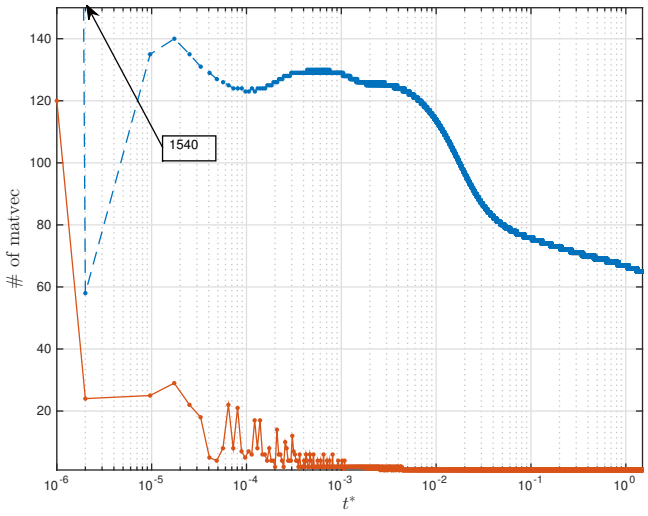

Figure 9: Impulsively started flat plate at Re = 1000: CPU cost to converge to a tolerance of 10^{-2}, Jacobi(1.9) (−−•−) and rBiCGStab(2) (−•−).

minimum number of iterations (i.e. resolutions of the Poisson equation) should be enforced to damp any spurious oscillation. Nevertheless, the flat plate problem indicates that sharp geometrical features and important unsteady effects exacerbate the differences between the two solvers, leading to even higher computational gains.

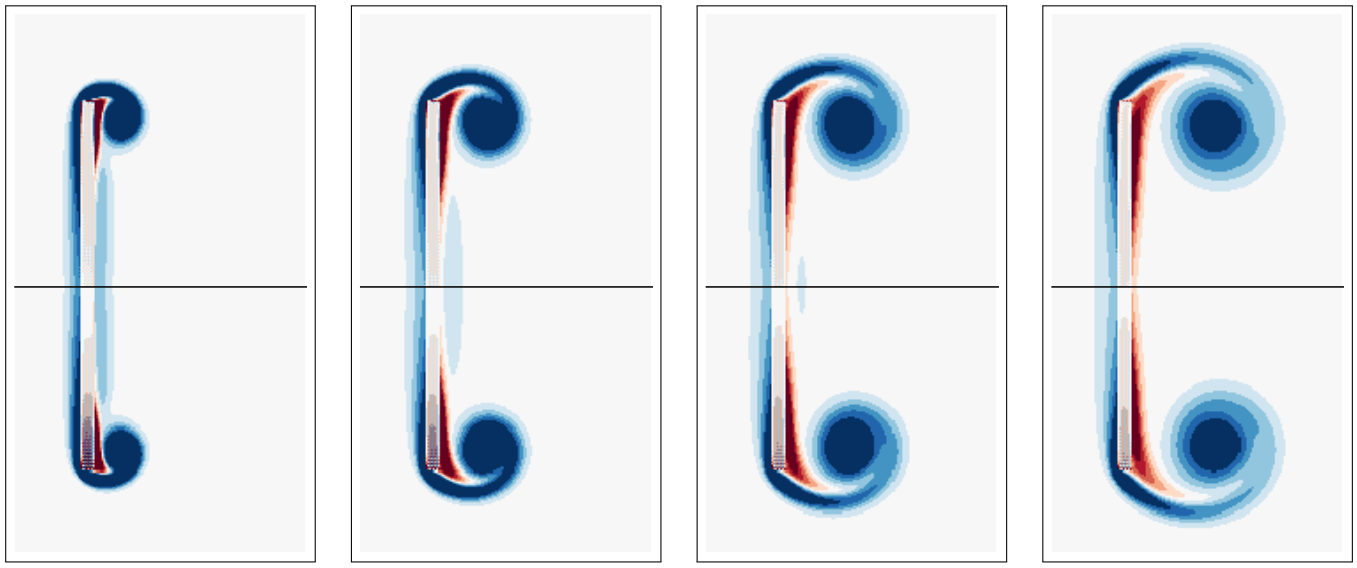

Figure 10: Impulsively started flat plate at Re = 1000: tip vorticity at $t^* = 0.02$, $t^* = 0.04$, $t^* = 0.06$, $t^* = 0.08$ (from left to right) obtained with rBiCGStab(2) (up) and Jacobi(1.9) (down) both with tolerance 10^{-2}.

5. Conclusion and perspectives

Penalization methods in vortex methods are very well suited for the use of recycling iterative solvers. The unknown penalization vorticity is restricted to the boundaries of the body, thus keeping the storage of previous solutions affordable. Solving for this penalization step is also potentially expensive as it entails an elliptic problem:

velocities induced by the correction vorticity have to be obtained, which amounts to solving a Poisson equation. Because the right-hand side changes only slightly between two time steps, previous solutions can be efficiently recycled. We developed the approach and motivated the adoption of the rBiCGStab (recycled Bi-Conjugate Gradient Stabilized) as the underlying solver for this application.

The resulting algorithm was assessed on benchmark problems: the flow past an impulsively started cylinder at low and moderate Reynolds numbers and the flow past an impulsively started flat plate. The convergence with respect to the external error is found to be twice as fast as that of the competing approach [16]. Recycling previous solutions also saves many iterations: it divides the cost of iterative penalization by a factor of $\simeq 50$ for the case of the cylinder ($\mathrm{Re} = 9500$) and $\simeq 120$ for the case of the flat plate ($\mathrm{Re} = 1000$). Furthermore, two previous solutions are found to be sufficient in order to maximize the computational gain and keep the memory overhead low enough to be usable for large problems. However, the higher computational costs encountered during the early phases of the impulsive start hint at the possible revision of this conclusion for flows with strong unsteadiness and/or fast body deformations: more stored previous solutions might be useful to reduce the cost.

In a directly related work, one could consider to study the use of a preconditioner, analyse the reason for a breakdown or take advantage of the recycling of eigenvectors in order to further improve the convergence. We could also develop a criterion in order to use an adaptative recycled subspace's dimension or transport the recycled solution to address accelerating problems. Finally, we note that the generalization of this work to 3D is relatively straightforward since the method considers the domain fields as one dimensional vectors. This is a topic of ongoing work.

Acknowledgements

We would like to acknowledge the helpful discussion with P. Van Dooren (ICTEAM: Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université catholique de Louvain). T. Gillis is supported by a FRIA grant from the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS). The present research also benefited from computational resources made available on the Tier-1 supercomputer of the Federation Wallonie-Bruxelles, infrastructure funded by the Walloon Region under the grant agreement $n°1117545$.

References

[1] P. Koumoutsakos, A. Leonard, High-resolution simulations of the flow around an impulsively started cylinder using vortex methods, Journal of Fluid Mechanics 296 (1) (1995) 1–38.

[2] G.-H. Cottet, P. Koumoutsakos, Vortex Methods, Theory and Practice, Cambridge University Press, 2000.

[3] G. S. Winckelmans, Encyclopedia of Computational Mechanics, chap. Vortex Methods, John Wiley & Sons, Ltd, ISBN 9780470091357, 129–153, 2004.

[4] P. Ploumhans, G. S. Winckelmans, Vortex Methods for High-Resolution Simulations of Viscous Flow Past Bluff Bodies of General Geometry, Journal of Computational Physics 165 (2) (2000) 354–406.

[5] P. Ploumhans, G. Winckelmans, Vortex Methods for Direct Numerical Simulation of Three-Dimensional Bluff Body Flows: Application to the Sphere at Re= 300, 500, and 1000, Journal of Computational Physics 178 (2) (2002) 427–463.

[6] G. H. Cottet, B. Michaux, S. Ossia, G. Vanderlinden, A comparison of spectral and vortex methods in three-dimensional incompressible flows., Journal of Computational Physics 175 (2) (2002) 702–712.

[7] G.-H. Cottet, P. Poncet, Advances in direct numerical simulations of 3D wall-bounded flows by Vortex-in-Cell methods, J. Comput. Phys. 193 (1) (2004) 136–158.

[8] M. Bergdorf, P. Koumoutsakos, A. Leonard, Direct numerical simulations of vortex rings at Re-Gamma=7500, Journal of Fluid Mechanics 581 (2007) 495–505.

[9] R. Cocle, G. Winckelmans, G. Daeninck, Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations, Journal of Computational Physics 227 (21) (2008) 9091–9120.

[10] P. Chatelain, A. Curioni, M. Bergdorf, D. Rossinelli, W. Andreoni, P. Koumoutsakos, Billion vortex particle Direct Numerical Simulations of aircraft wakes, Computer Methods in Applied Mechanics and Engineering 197 (13) (2008) 1296–1304.

[11] Y. Marichal, P. Chatelain, G. Winckelmans, An immersed interface solver for the 2-D unbounded Poisson equation and its application to potential flow, Computers & Fluids 96 (2014) 76–86.

[12] Y. Marichal, P. Chatelain, G. Winckelmans, Immersed interface interpolation schemes for particle–mesh methods, Journal of Computational Physics 326 (2016) 947 – 972.

[13] P. Angot, C. Bruneau, P. Fabrie, A penalization method to take into account obstacles in incompressible viscous flows, Numerische Mathematik 81 (1999) 497–520.

[14] N. K.-R. Kevlahan, J.-M. Ghidaglia, Computation of turubulent flow past an array of cylinders using a spectral method with Brinkman penalization, European Journal of Mechanics - B/Fluids 20 (3) (2001) 333–350.

[15] C. Mimeau, F. Gallizio, G.-H. Cottet, I. Mortazavi, Vortex penalization method for bluff body flows, International Journal for Numerical Methods in Fluids 79 (2) (2015) 55–83, fld.4038.

[16] M. M. Hejlesen, P. Koumoutsakos, A. Leonard, J. H. Walther, Iterative Brinkman penalization for remeshed vortex methods, Journal of Computational Physics 280 (2015) 547–562.

[17] J. T. Rasmussen, G.-H. Cottet, J. H. Walther, A multiresolution remeshed Vortex-In-Cell algorithm using patches, Journal of Computational Physics 230 (2011) 6742–6755.

[18] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM Journals, 2003.

[19] Y. Saad, M. H. Schultz, GMRes: A Generalized Minimal Residual Algorithm For Solving NonSymmetric Linear Systems, SIAM J. Sci. Stat. Comput. 7 (3).

[20] P. Sonneveld, CGS, a Fast Lanczos-type Solver for Nonsymmetric Linear Systems, SIAM J. Sci. Stat. Comput. 10 (1) (1989) 36–52.

[21] H. A. van der Vorst, BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems, SIAM J. Sci. Stat. Comput. 13 (2) (1992) 631–644.

[22] E. de Sturler, Nested Krylov methods based on GCR, Journal of Computational and Applied Mathematics 67 (1996) 15–41.

[23] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johson, S. maiti, Recycling Krylov subspaces for sequences of linear systems, SIAM Journal on Scientific Computing 28 (5) (2006) 1651–1674.

[24] R. B. Morgan, GMRes with deflated restarting, SIAM Journal on Scientific Computing 24 (1) (2002) 20–37.

[25] M. H. Carpenter, A General Algorithm for Reusing Krylov Subspace Informatin., Technical Memorandum NASA/TM–2010–216190, Langley Research Center, 2010.

[26] K. Ahuja, P. Benner, E. de Sturler, L. Feng, Recycling BiCGStab with an application to parametric model order reduction, SIAM Journal on Scientific Computing 37 (5) (2015) S429–S446.

[27] E. de Sturler, Truncation Strategies for optimal Krylov subspace methods, SIAM Journal on Numerical Analysis 36 (3) (1999) 864–889.

[28] A. Amritkar, E. Sturler, K. Swirydowicz, D. Tafti, K. Ahuja, Recycling Krylov subspaces for CFD applications and a new hybrid recycling solver, Journal of Computational Physics 303 (2015) 222–237.

[29] M. Coquerelle, G. H. Cottet, A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies, Journal of Computational Physics 227 (21) (2008) 9121–9137.

[30] R. Hockney, J. Eastwood, Computer Simulation using Particles, Taylor & Francis, Inc. Bristol, PA, USA, 1988.

[31] P. Chatelain, P. Koumoutsakos, A Fourier-based elliptic solver for vortical flows with periodic and unbounded directions, Journal of Computational Physics 229 (7) (2010) 2425–2431.

[32] M. Gazzola, P. Chatelain, W. M. van Rees, P. Koumoutsakos, Simulations of single and multiple swimmers with non-divergence free deforming geometries, Journal of Computational Physics 230 (19) (2011) 7093–7114.

[33] D. Rossinelli, M. Bergdorf, G.-H. Cottet, P. Koumoutsakos, GPU Accelerated Simulations of Bluff Body Flows Using Vortex Particle Methods, J. Comput. Phys. 229 (9) (2010) 3316–3333.

[34] C. Lanczos, Solution of systems of linear equations by minimized iterations, Journal of Research of the National Bureau of Standards 49 (1952) 33–53.

[35] F. Noca, D. Shiels, D. Jeon, A comparison of methods for evaluating time-dependent fluid dynamic forces on bodies, using only velocity fields and their derivatives, Journal of Fluids and Structures 13 (5) (1999) 551–578.

[36] M. Bar-Lev, H. T. Yang, Initial flow field over an impulsively started circular cylinder, Journal of Fluid Mechanics 72 (1975) 625–647.

[37] Y. Marichal, An immersed interface vortex particle-mesh method, Ph.D. thesis, Université catholique de Louvain, 2014.