HAL
open science

Iterative solver approach for turbine interactions: application to wind or marine current turbine farms

Paul Mycek, Grégory Pinon, Corentin Lothodé, Alexandre Dezotti, Clément Carlier

To cite this version:

Paul Mycek, Grégory Pinon, Corentin Lothodé, Alexandre Dezotti, Clément Carlier. Iterative solver approach for turbine interactions: application to wind or marine current turbine farms. Applied Mathematical Modelling, 2017, 41, pp.331 - 349. 10.1016/j.apm.2016.08.027 . hal-01919332

HAL Id: hal-01919332

https://normandie-univ.hal.science/hal-01919332v1

Submitted on 5 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL
open science

Iterative solver approach for turbine interactions: application to wind or marine current turbine farms

Paul Mycek, Grégory Pinon, Corentin Lothodé, Alexandre Dezotti, Clément Carlier

To cite this version:

Paul Mycek, Grégory Pinon, Corentin Lothodé, Alexandre Dezotti, Clément Carlier. Iterative solver approach for turbine interactions: application to wind or marine current turbine farms. Applied Mathematical Modelling, Elsevier, 2017, 41, pp.331 - 349. 10.1016/j.apm.2016.08.027 . hal-01919332

HAL Id: hal-01919332

https://hal-normandie-univ.archives-ouvertes.fr/hal-01919332

Submitted on 5 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Iterative solver approach for turbine interactions: application to wind or marine current turbine farms

Paul Mycek[a,*], Grégory Pinon[b], Corentin Lothodé[c,d], Alexandre Dezotti[e], Clément Carlier[b,f]

[a] MEMS department, Duke University, 144 Hudson Hall, Box 90300, Durham, NC 27708, USA
[b] Laboratoire Ondes et Milieux Complexes, UMR 6294, CNRS – Université du Havre, 53, rue de Prony, BP 540, Le Havre Cedex F-76058, France
[c] Laboratoire d'Optimisation et Fiabilité en Mécanique des Structures, EA 3828, INSA de Rouen, Avenue de l'Université, BP 08, Saint-Etienne-du-Rouvray F-76801, France
[d] K-Epsilon WTC-Bâtiment 1 - Entrée E, 1300 Route des Crêtes, F-06560 Valbonne, France
[e] Department of Mathematical Sciences, University of Liverpool, Liverpool L69 7ZL, United Kingdom
[f] IFREMER, Marine Structures Laboratory, 150, quai Gambetta, BP 699, Boulogne-Sur-Mer F-62321, France

ARTICLE INFO

Article history:
Received 24 July 2015
Revised 8 August 2016
Accepted 24 August 2016
Available online 30 August 2016

Keywords:
Iterative solver
Bi-GCSTAB
Preconditioner
Lagrangian vortex method
Wind turbine
Marine current turbine

ABSTRACT

This paper presents a numerical investigation for the computation of wind or marine current turbines in a farm. A 3D unsteady Lagrangian vortex method is used together with a panel method in order to take into account for the turbines. In order to enforce the boundary condition onto the panel elements, a linear matrix system is defined. Solving general linear matrix systems is a topic with important scientific literature. But the main concern here is the application to a dedicated matrix which is non-sparse, non-symmetric, neither diagonally dominant nor positive-definite. Several iterative approaches were tested and compared. But after some numerical tests, a Bi-CGSTAB method was finally chosen. The main advantage of the presented method is the use of a specific preconditioner well suited for the desired application. The chosen implementation proved to be very efficient with only 3 iterations of our preconditioned Bi-CGSTAB algorithm whatever the turbine geometrical configuration. Although developed for wind or marine turbines, the proposed algorithm is absolutely not restricted to these cases, and can be applied to many others. At the end of the paper, some applications (specifically, wake computations) in a farm are presented, along with a quantitative assessment of the computational time savings brought by the iterative approach.

1. Introduction

Turbines, whether they are wind or water marine current turbines, represent a growing interest in the scientific community for energy and environmental engineering. From an historical point of view, the first studies were performed on wind turbines and recently, a similar research procedure is being developed regarding marine hydro-kinetic or marine current turbines. Furthermore, computational fluid dynamics (CFD) has been, since the beginning of these studies, a major concern for

* Corresponding author.
E-mail addresses: paul.mycek@duke.edu, paul.mycek@gmail.com (P. Mycek).

http://dx.doi.org/10.1016/j.apm.2016.08.027

first aerodynamics and secondly hydrodynamics. More and more physical phenomena are being taken into account leading to an increase in the complexity of the developed numerical methods. An extremely detailed review of numerical methods dedicated to wind turbine aerodynamics and aeroelasticity was carried out by Hansen et al. [1], and a more recent one by Miller et al. [2]. A similar review for marine turbine applications was also published recently [3].

Several computational techniques exist with increasing complexity, from a classical blade element momentum (BEM) theory to a fully 3D unsteady Navier–Stokes formulation, including boundary layer treatment around the blades. Unfortunately, for the interaction of several turbines within a farm, this last approach is not really affordable at the present time; although some researchers took up the challenge [4] with impressive results. The present paper aims at describing a numerical implementation for the computation of marine current turbine hydrodynamics [5,6]. To some extent, the developed software is similar to those developed by Baltazar et al. [7] or McCombes et al. [8], also for marine current turbine applications. However, there is no real restriction to marine current turbine hydrodynamics and it may largely be used in wind energy applications [9–13]. Here, both the performances (power and thrust coefficients) and the wake are considered in an unsteady Lagrangian vortex method [14]. The blades are taken into account with a panel method using a Kutta condition for the emission of vortex particles [15]. The major advantage of such methods (that is to say panel method with free vortex blobs) is that it does not require any 3D meshing of the fluid domain, the only mesh being the one used for the discretisation of the blades. Therefore, there is no special treatment if one wants to compute several turbines in interaction, whereas classical Eulerian methods would require sophisticated meshes, probably with several rotating parts if several turbines are considered [16]. However, integral panel methods impose the resolution of a linear matrix system at each unsteady time step. The present paper aims at describing an enhanced formulation for rapid solving of such matrix systems in the case of several turbines in a farm.

Generally speaking, in Lagrangian vortex methods, should one want to treat boundary conditions, a system of linear equations appears. The present implementation [5,6] does not take dynamic stall into account, but improvements like those suggested by Voutsinas and Riziotis [10,11] make it possible to do so. In this formulation, the resolution of a matrix linear system is also required. For instance, no-slip boundary conditions can be considered with Lagrangian vortex method. This approach, developed in 2D by Ploumhans et al. [17], followed by its 3D version [18], also uses a matrix system issuing from the surface discretisation. Boundary conditions may also be enforced through Immersed Boundary (IB) and a version dedicated to velocity-vorticity formulation was proposed by Poncet [19]. To the authors' knowledge, such an implementation has never been applied to (marine or wind) turbine computations, mainly owing to their computational cost at such Reynolds numbers. However, Poncet's Immersed Boundary method [19] was applied to model the flow around a plane, which tends to give confidence in such possibilities. In some sense, the proposed numerical treatment for the resolution of the matrix system for several turbines in interaction may be extended to all these approaches [5,6,10,11,17–19]. Moreover, the presented method is not limited to turbine interactions but may be applied to the computation of several non-deformable moving objects.

In the above mentioned studies, the matrix system basically represents geometrical compounds of the equations to solve. In most cases, when treating one single rigid structure, the matrix is constant over time. In a former study [5], when computing a single rigid turbine, the matrix was a time-constant and matrix inversion was cost effective. A parallel Gauss–Jordan method was then implemented; the matrix inverse was computed once and for all at the beginning of the simulation and stored prior to the unsteady iterations. At each time step, a simple matrix-vector multiplication was used. When computing turbine interactions (starting by two turbines only [6]), as the matrix is no longer a time-constant owing to the relative motion of the two turbines (see Fig. 5), matrix inversion at each time step becomes too expensive in terms of CPU time. Following the literature, we consider iterative methods such as Jacobi, Gauss–Seidel, Conjugate–Gradient (CG) and its variants. In order to choose the best implementation, a matrix characterisation study is performed in Section 3. Important literature exists on CG-class methods with essentially applications to large matrices (up to billions of elements [20]). To have a better understanding of CG methods, the reader may refer to the intuitive and comprehensive lecture note by Shewchuk [21]. The implemented Bi-CGSTAB comes from van der Vorst [22] and the convergence is compared with two other methods, namely an iterative Jacobi method and a classical Conjugate-Gradient. The Bi-CGSTAB appears to be much more efficient with appropriate preconditioning. A suitable and efficient preconditioner, which takes advantage of the block structure of the involved matrix, is derived and appears to considerably accelerate the convergence of the system solve.

The paper is organised as follows. First, the numerical method for flow simulation is presented in Section 2 as in references [5,6]. Then, several matrices are well characterised in Section 3 in order to explain the choice of the Bi-CGSTAB. A specific preconditioner is presented and convergence is analysed in Section 3.2 with and without the proposed preconditioner. Last, in Section 4, the proposed approach is applied to the simulation, in terms of wake characterisation, of elementary interactions between marine current turbines in a farm. Computational times are presented and compared with those obtained using a direct solver (*i.e.* direct matrix inversion).

2. Description of the numerical method

The following paragraphs give the mathematical background regarding the governing equations for the modelling of turbine farms by means of the proposed approach. One can also refer to [5] for further details. The set-up consists of an exterior fluid domain $\mathcal{V}$ with moving boundaries $\mathcal{S}$. Here, the boundaries $\mathcal{S}$ correspond to the surfaces of the turbine blades and hub.

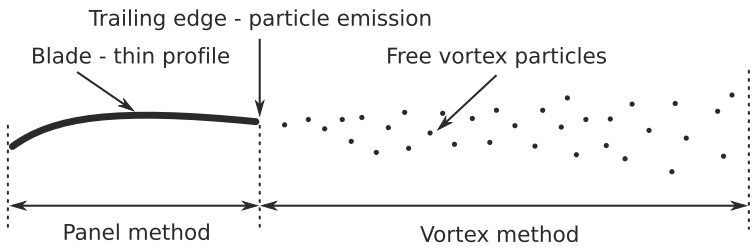


Fig. 1. Schematic cross-sectional view of a blade profile showing the respective locations of the panel method and of the particle method, respectively.

The flow is governed by the incompressible Navier-Stokes equations written in velocity-vorticity formulation ($\mathbf{u}$, $\boldsymbol{\omega}$):

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \nu\boldsymbol{\Delta}\boldsymbol{\omega}, \quad \boldsymbol{\omega} = \nabla\wedge\mathbf{u}, \tag{1}$$

together with the continuity equation, which reduces, in an incompressible fluid, to a divergence-free velocity field:

$$\operatorname{div}\mathbf{u} = \nabla\cdot\mathbf{u} = 0. \tag{2}$$

The operator $\mathrm{D}/\mathrm{D}t$ stands for the material derivative and ν represents the fluid viscosity. The velocity $\mathbf{u}$ is decomposed according to the Helmholtz decomposition:

$$\mathbf{u} = \mathbf{u}^{\boldsymbol{\psi}} + \mathbf{u}^{\phi} + \mathbf{u}^{\infty}, \tag{3}$$

where the vector field $\mathbf{u}^{\boldsymbol{\psi}}$ is the rotational part of the velocity field, the vector field $\mathbf{u}^{\boldsymbol{\phi}}$ is the potential velocity field and $\mathbf{u}^{\infty}$ is the velocity field representing the marine current inflow, assumed here to be constant and uniform. The first two velocity fields are defined from the vector potential $\boldsymbol{\psi}$ and the scalar potential ϕ as follows:

$$\mathbf{u}^{\boldsymbol{\psi}} = \nabla\wedge\boldsymbol{\psi}, \quad \mathbf{u}^{\phi} = \nabla\phi. \tag{4}$$

These potentials, owing to Eq. (2), satisfy the following equations:

$$\boldsymbol{\Delta}\boldsymbol{\psi} = -\boldsymbol{\omega}, \tag{5}$$

$$\Delta\phi = 0. \tag{6}$$

From Eq. (5), the rotational part of the velocity field is given by the Biot–Savart law:

$$\mathbf{u}^{\boldsymbol{\psi}}(M) = \frac{1}{4\pi}\int_{\mathcal{V}} \mathbf{K}(\mathbf{MM}')\wedge\boldsymbol{\omega}(M')\,\mathrm{d}\nu(M'), \tag{7}$$

where M and M' are two points in $\mathcal{V}$ and $\mathbf{K}(\mathbf{x}) = \mathbf{x}/|\mathbf{x}|^3$ denotes the Biot–Savart kernel. From Eq. (6), the potential velocity $\mathbf{u}^{\boldsymbol{\phi}}$ can be expressed as:

$$\mathbf{u}^{\phi}(M) = \frac{1}{4\pi}\nabla_M \iint_{\mathcal{S}} \mu(P)\frac{\mathbf{MP}\cdot\mathbf{n}(P)}{|\mathbf{MP}|^3}\,\mathrm{d}s(P), \tag{8}$$

where $\mathrm{d}s$ is the surface measure on $\mathcal{S}$ and $\mathbf{n}(P)$ stands for the vector normal to the surface $\mathcal{S}$ at a point P.

The function μ represents a distribution of normal dipoles on the turbines blades surfaces $\mathcal{S}$, which is *a priori* unknown. It is determined by the boundary condition on $\mathcal{S}$. For any point P on $\mathcal{S}$, a slip velocity condition is enforced as follows:

$$\mathbf{u}^{\phi}(P)\cdot\mathbf{n}(P) = \left[\mathbf{u}^{\mathrm{rot}}(P) - \mathbf{u}^{\boldsymbol{\psi}}(P) - \mathbf{u}^{\infty}(P)\right]\cdot\mathbf{n}(P), \tag{9}$$

where the vector field $\mathbf{u}^{\mathrm{rot}}$ represents the velocity imposed by the rotation of the blade surfaces $\mathcal{S}$. Finally, the diffusion term of Eq. (1), $\nu\boldsymbol{\Delta}\boldsymbol{\omega}$, can be modelled by means of the broadly-used particle strength exchange (PSE) method [23–26]. Alternatively, the diffusion velocity method (DVM) [27–31] may be used instead. Both of these methods may be modified in order to incorporate a LES turbulence model [5,32].

The discretisation process consists of two parts: first the blades surfaces $\mathcal{S}$ are discretised using an integral panel method; and second, the fluid domain $\mathcal{V}$ is discretised into free vortex particles, also called free vortex blobs. Vortex particles are emitted at trailing edge of the blades according to a Kutta condition around these lifting surfaces. A sketch of the method is depicted in Fig. 1 and one can also refer to [5,9,33].

2.1. The meshing of the blades and the potential velocity

Let N denote the total number of surface mesh elements, the surface $\mathcal{S}$ is then decomposed into N polygonal elements $\{S_p\}_{p=1}^{N}$. The distribution of normal dipoles μ is assumed to be constant on each element. Let μ_p denote the value of μ on S_p, and let $\mathbf{n}_p$ denote the unit normal vector. In the present study, only quadrangular elements are considered but there is no

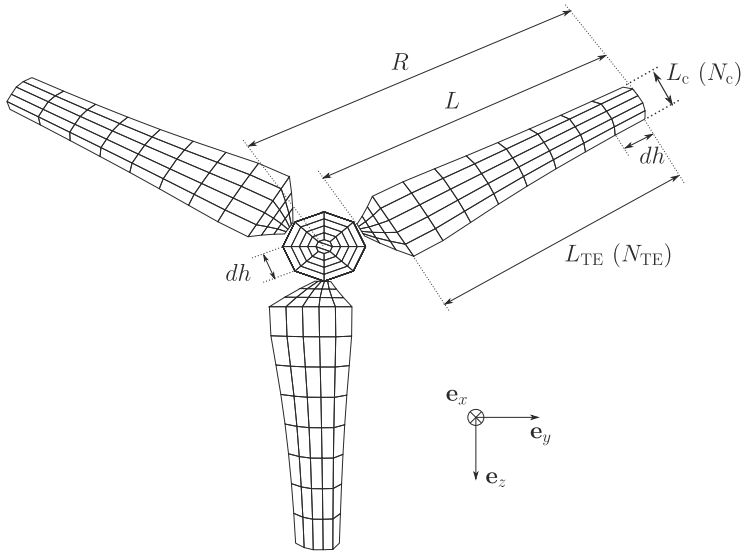


Fig. 2. Description of the meshing parameters on a single turbine model reproduced from [5]. The different meshing characteristics are given in Table 1 of Section 3.

restriction on the mesh element type and one may choose triangles, pentagons, etc., provided that the following equations are modified accordingly. A typical turbine mesh, reproduced from [5], is illustrated on Fig. 2.

From these assumptions, it follows that Eq. (8) can be recast, for $M \in \mathcal{V}$, as

$$\mathbf{U}^{\phi}(M) = \sum_{p=1}^{N} \frac{\mu_p}{4\pi} \nabla_M \iint_{S_p} \frac{\mathbf{MP} \cdot \mathbf{n}(P)}{|\mathbf{MP}|^3} \, \mathrm{d}s(P). \tag{10}$$

It then follows from Stokes' formula [34–37] that

$$\mathbf{U}^{\phi}(M) = \frac{1}{4\pi} \sum_{p=1}^{N} \mu_p \sum_{k=0}^{3} \int_{\ell_p^k} \frac{\mathbf{MP}}{|\mathbf{MP}|^3} \wedge \mathbf{dP}, \tag{11}$$

where $\ell_p^0, \ell_p^1, \ell_p^2, \ell_p^3$ are the four edges of the element S_p, whose corresponding vectors may be oriented according to the direction of the normal $\mathbf{n}_p$.

2.2. The vortex particles and the rotational velocity

The vortical fluid domain (see right hand side of Fig. 1) is discretised using particles (or blobs). The fluid properties, and particularly its vorticity field $\boldsymbol{\omega}$, are represented by a finite family of vortex particles $\{\mathcal{P}_i\}_{i=1}^{N_\mathrm{P}}$. The use of particles amounts to a Lagrangian aspect of the flow computation, where the particles are seen as material parts of the fluid that evolve according to the fluid motions. In addition, the number of particles, denoted by N_P, may vary with time.

For each particle $\mathcal{P}_i$, the following quantities are defined:

- $V_i = \int_{\mathcal{P}_i} \mathrm{d}\nu$ is the volume of particle $\mathcal{P}_i$;
- $\mathbf{X}_i = \int_{\mathcal{P}_i} \mathbf{x} \, \mathrm{d}\nu / V_i$ is its position; and
- $\boldsymbol{\Omega}_i = \int_{\mathcal{P}_i} \boldsymbol{\omega} \, \mathrm{d}\nu$ is its vortical weight.

Each particle $\mathcal{P}_i$ also has its own velocity $\mathbf{U}_i = \mathbf{U}(\mathbf{X}_i)$, which is decomposed as in Eq. (3) into the potential component $\mathbf{U}_i^{\phi} = \mathbf{U}^{\phi}(\mathbf{X}_i)$ (Eq. (11)), the far-field velocity $\mathbf{U}^{\infty}$, and the rotational part $\mathbf{U}_i^{\boldsymbol{\psi}} = \mathbf{U}^{\boldsymbol{\psi}}(\mathbf{X}_i)$, as mentioned previously. This last component of the velocity field can be discretised from Eq. (7) by:

$$\mathbf{U}^{\boldsymbol{\psi}}(M) = \sum_{j=1}^{N_\mathrm{P}} \mathbf{K}_{\varepsilon}(\mathbf{MX}_j) \wedge \boldsymbol{\Omega}_j, \tag{12}$$

where the kernel $\mathbf{K}_{\varepsilon}$ is a smoothed version of the Biot–Savart kernel $\mathbf{K}$ in Eq. (7), assumed to converge to $\mathbf{K}$ when the smoothing parameter ε tends to 0 [38,39]. The Helmholtz decomposition of Eq. (3) can now be expressed in a discretised form as follow:

$$\mathbf{U}_i = \mathbf{U}_i^{\boldsymbol{\psi}} + \mathbf{U}_i^{\phi} + \mathbf{U}^{\infty}. \tag{13}$$

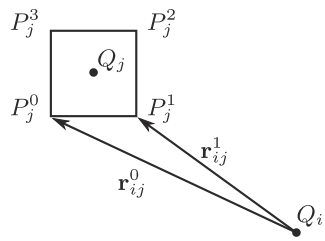


Fig. 3. Representation of the $\mathbf{r}_{ij}^k$ (with $k=0$) for the evaluation of the curvilinear integral in Eq. (17). The edges of panel S_j are defined as $\ell_j^k = [P_j^k, P_j^{k+1}]$, where $k+1$ is taken modulo 4.

Because the fluid is incompressible, the volume V_i of each particle is constant. On the contrary, the position, the vortical weight, as well as the velocity of the particles depend on the time t. For clarity, the dependence of these quantity on t is omitted hereafter. Owing to the Lagrangian frame, the positions of the particles evolve according to the velocity as follows:

$$\frac{\mathrm{d}\mathbf{X}_i}{\mathrm{d}t} = \mathbf{U}_i. \tag{14}$$

The equation satisfied by the vortical weight $\mathbf{\Omega}_i$ can be derived from the vorticity transport Eq. (1):

$$\frac{\mathrm{d}\mathbf{\Omega}_i}{\mathrm{d}t} = (\mathbf{\Omega}_i \cdot \nabla)\mathbf{U}_i + V_i(\nu \mathbf{\Delta\omega})_{\mathbf{x}=\mathbf{X}_i}. \tag{15}$$

This requires the evaluation of derivatives of the discretised velocity and vorticity fields. The discretised tensor field $\nabla\mathbf{U}^\phi$ can be evaluated by differentiating Eq. (11), while the gradient of $\mathbf{U}^\psi$ can be obtained by differentiating the kernel $\mathbf{K}_\varepsilon$. The diffusion term $(\nu\mathbf{\Delta\omega})_{\mathbf{x}=\mathbf{X}_i}$ is discretised using the Particle Strength Exchange method [23–26]. Finally, a second order Runge–Kutta scheme is used for the time integration of the system of ordinary differential equations governing the evolution of the position (Eq. (14)) and vortical weight (Eq. (15)).

2.3. The influence matrix

The boundary condition (9) leads to a system of linear equations. Because the normal dipole distribution on $\mathcal{S}$ is assumed to be uniform on each panel (piecewise constant), it can be represented by a vector of N scalar values $\boldsymbol{\mu} = (\mu_j)_{j=1}^N$. Then the boundary condition takes the form of a system of N linear equations, $A\boldsymbol{\mu} = \mathbf{b}$, whose solution $\boldsymbol{\mu}$ enforces the boundary condition at the centres of the panels. The right hand side vector $\mathbf{b} = (b_i)_{i=1}^N$ is given by:

$$b_i = \left[\mathbf{U}^{\mathrm{rot}}(Q_i) - \mathbf{U}^\psi(Q_i) - \mathbf{U}^\infty\right] \cdot \mathbf{n}_i, \tag{16}$$

where Q_i denotes the centre of panel S_i (see Fig. 3) and the velocity $\mathbf{U}^\psi(Q_i)$ is evaluated through Eq. (12). From Eq. (11), it follows that the so-called *influence matrix*, $A = (a_{ij})_{i,j=1}^N$, is given by:

$$a_{ij} = \frac{\mathbf{n}_i}{4\pi} \cdot \sum_{k=0}^{3} \int_{\ell_j^k} \frac{\mathbf{Q}_{iP}}{|\mathbf{Q}_{iP}|^3} \wedge \mathbf{dP}. \tag{17}$$

It is possible to explicitly compute the value of the integrals $\int_{\ell_j^k} \mathbf{Q}_{iP}/|\mathbf{Q}_{iP}|^3 \wedge \mathbf{dP}$. Indeed, if we define $\mathbf{r}_{ij}^k = \mathbf{Q}_{iP_j^k}$, as depicted on Fig. 3, then we have [34,40,41]:

$$a_{ij} = \frac{\mathbf{n}_i}{4\pi} \cdot \sum_{k=0}^{3} \left(|\mathbf{r}_{ij}^k| + |\mathbf{r}_{ij}^{k+1}|\right)\left(1 - \frac{\mathbf{r}_{ij}^k \cdot \mathbf{r}_{ij}^{k+1}}{|\mathbf{r}_{ij}^k|\,|\mathbf{r}_{ij}^{k+1}|}\right) \frac{\mathbf{r}_{ij}^k \wedge \mathbf{r}_{ij}^{k+1}}{|\mathbf{r}_{ij}^k \wedge \mathbf{r}_{ij}^{k+1}|^2}, \tag{18}$$

where $k+1$ is taken modulo 4. The discrete potential velocity given by Eq. (11) is evaluated using the same geometric representation of the curvilinear integral, Q_i being replaced by any point M in $\mathcal{V}$.

The coefficient a_{ij} of the matrix A represents the influence of element S_j onto element S_i, which is generally different from the influence a_{ji} of element S_i onto element S_j, owing to the meshing process (Section 3.1). As a consequence, A is generally non-symmetric. When a single turbine is considered, the matrix A basically represents the influence of a single turbine on itself. Fig. 4 gives a schematic view of such a problem involving one single turbine. As illustrated, because the turbine is assumed to be undeformable, the rotation $\dot{\theta}$ does not change the coefficients of A defined in Eq. (18). This is generally true for any isometric transformation, as long as the solid body is undeformable. As a matter of fact, the dot and cross products are invariant by isometric transformation, and A is thus constant with respect to time. For example, the influence of a given panel S_i on its own centre point, like the interaction referred to as (1) on Fig. 4, remains unchanged by the rotation. Likewise, the influence of a panel S_j on the centre point Q_k of another panel S_k pertaining to the same body

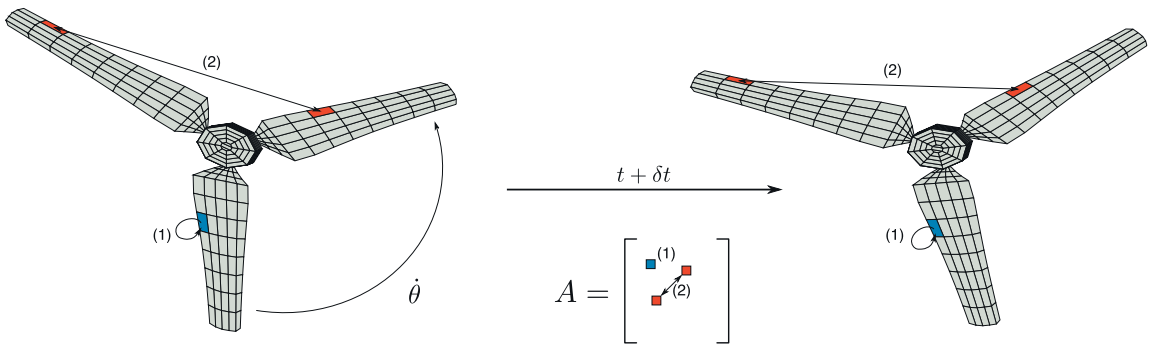


Fig. 4. Schematic view of the single turbine case.

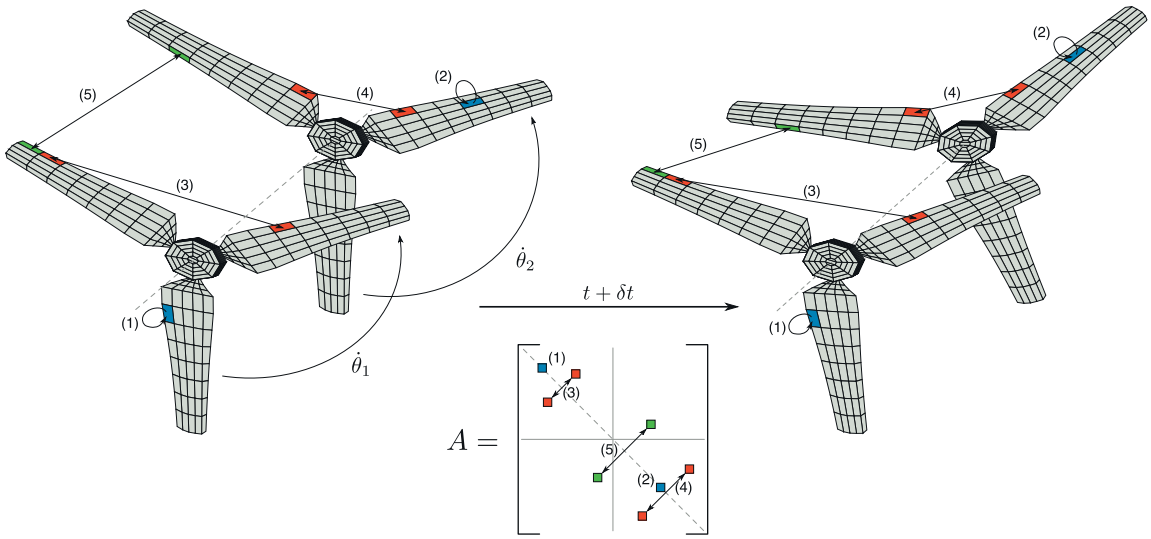


Fig. 5. Schematic view of the twin-turbines case ($n = 2$).

(subject to the same rotation) does not change, as illustrated by interaction (2). This is the reason why direct inversion was preferred in previous works where a single turbine was considered [5].

On the contrary, when $n > 1$ turbines are concerned, the global matrix A is no longer constant over time. Each individual turbine $\mathcal{T}_i$ (for $i = 1, \ldots, n$) is discretised by a contiguous subfamily of the family of panels $\{S_p^{(i)}\}_{p=1}^{N_i} \subset \{S_p\}_{p=1}^{N}$. The matrix A then has a natural block structure:

$$A = \begin{bmatrix} [A_{11}] & [A_{12}] & \cdots & [A_{1n}] \\ [A_{21}] & [A_{22}] & \cdots & [A_{2n}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{n1}] & [A_{n2}] & \cdots & [A_{nn}] \end{bmatrix}. \tag{19}$$

To each pair of turbines $(\mathcal{T}_i, \mathcal{T}_j)$ corresponds a block $[A_{ij}]$ of the influence matrix containing the influence of the turbine $\mathcal{T}_i$ on the turbine $\mathcal{T}_j$. In particular, each diagonal block $[A_{ii}]$ is a square block and corresponds to the well-posed problem of the influence of a single turbine on itself. As a consequence, diagonal blocks $[A_{ii}]$ do not change with time, owing to the arguments mentioned above. Fig. 5 depicts the simplest case of $n = 2$ turbines. As an illustration, the schematic intra-turbine interactions (1) and (3), as well as (2) and (4) depicted on Fig. 5 remain unmodified with the rotation of the turbines and therefore $[A_{11}]$ and $[A_{22}]$ are constant blocks. Conversely, one can see that inter-turbines interactions are modified at each time step, for instance the schematic interaction number (5) on Fig. 5. Consequently, any extra-diagonal block $[A_{ij}]$, that is with $i \neq j$, corresponds to the interactions between distinct turbines. Even if the rotation speeds $\dot{\theta}_1$ and $\dot{\theta}_2$ were the same, the elements in the extra-diagonal blocks would change with time.[1]

[1] One particular scenario is when the rotation speeds are the same and the turbines are perfectly aligned with the rotation axis, in which case they can be considered as a single body subject to the same rotation, and then the matrix remains constant.

Table 1
Mesh description for different values of $\varepsilon = \kappa\, dh$, where ε denotes the smoothing parameter (see Eq. (12)), dh represents the characteristic mesh size (see Fig. 2), and κ is the overlapping ratio. N represents the total number of mesh elements. For each turbine, the asymmetry of the corresponding matrix A is characterised by β (Eq. (20)). The number γ of negative eigenvalues of $(1/2)(A + A^T)$ is also indicated.

ε	N_c	N_{TE}	N_c^{hub}	N_{TE}^{hub}	N	β	γ
0.200	5	5	6	6	182	0.0629	0
0.150	5	7	6	8	233	0.0903	0
0.100	5	11	6	12	339	0.1335	0
0.075	5	15	6	16	446	0.1594	1
0.050	5	23	6	24	666	0.1765	1
0.200	10	5	12	6	338	0.0284	0
0.150	10	7	12	8	431	0.0397	0
0.100	10	11	12	12	621	0.0626	0
0.075	10	15	12	16	812	0.1110	0
0.050	10	23	12	24	1200	0.1765	1
0.200	15	5	18	6	494	0.0281	0
0.150	15	7	18	8	629	0.0300	0
0.100	15	11	18	12	903	0.0398	0
0.075	15	15	18	16	1178	0.0704	1
0.050	15	23	18	24	1734	0.1544	1
0.200	5	5	58	6	488	0.0624	0
0.150	5	7	58	8	641	0.1000	2
0.100	5	11	58	12	964	0.1451	1
0.075	5	15	58	16	1278	0.1594	1
0.050	5	23	58	24	1914	0.1766	2

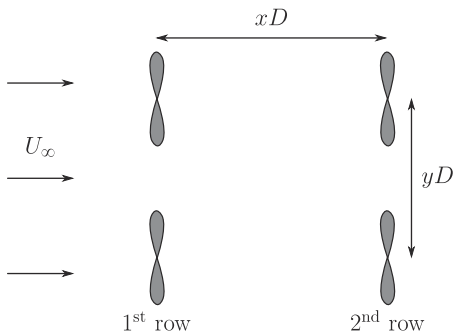
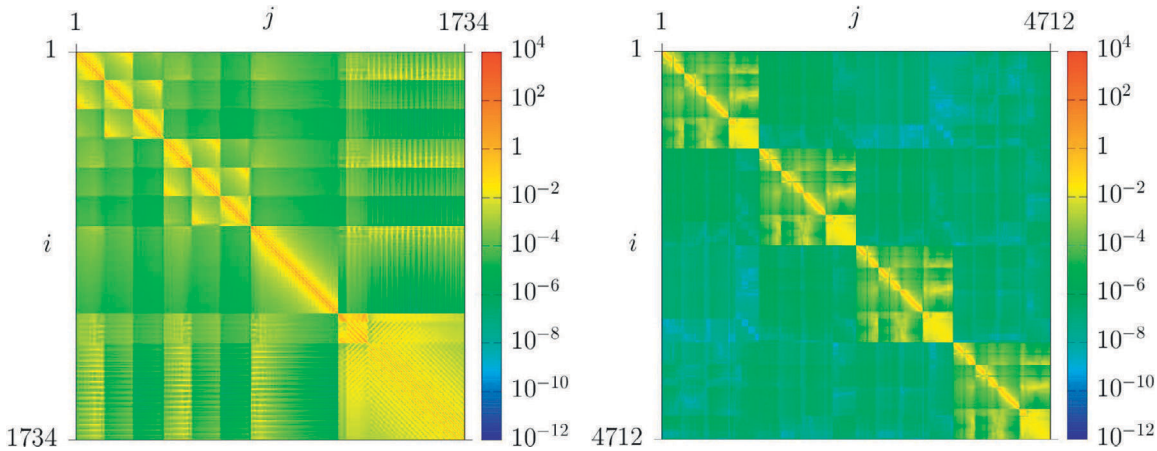


Fig. 6. Schematic top view of an aligned $A^{2\times 2}_{xD,yD}$ layout.

2.4. Turbine definition, meshes and multi-turbines layout

In the present study, the numerical configuration described in Ref. [5] is considered, with the "IFREMER-LOMC" blades. The reader may refer to [5,42] for a detailed description of the turbine we are modelling in the present paper. Because the computations are run dimensionless, the turbine radius is basically $R = 1$ and the characteristic mesh size for the trailing edge is defined by dh (see Fig. 2 for details). The overlapping parameter $\kappa = \varepsilon/dh$, defined as the ratio between the smoothing parameter ε (see Eq. (12)) and the characteristic mesh size dh (see Fig. 2), is kept approximately constant. The mesh parameters use subsequently in this paper are described in Table 1. N_{TE} (*resp.* N_{TE}^{hub}) represents the discretisation of a blade's trailing edge (*resp.* of the hub's trailing edge). N_c (*resp.* N_c^{hub}) represents the discretisation along the chord of the blade (*resp.* along the hub's length). N represents the total number of mesh elements and is greater than $N_c \times N_{TE} + N_c^{hub} \times N_{TE}^{hub}$ owing to the discretisation of the hub's front cross-sectional area. Each mesh is referenced using a label of the form $N_c \times N_{TE}$. The last five meshes, however, correspond to a slightly different geometry, with a longer hub, while the blade geometry and meshing remain the same. The meshes corresponding to those geometries will thus be labeled $N_c \times N_{TE}{}^*$ to distinguish them from their short hub counterpart. For instance, the very first mesh in Table 1 is labeled 5×5, while the very last one is labeled $5 \times 23^*$. Considering a single turbine configuration, with the meshes presented in Table 1, the size of the influence matrix A ranges approximately from 200×200 to $2{,}000 \times 2{,}000$. For a 10-turbines configuration with the finer turbine discretisation, a $20{,}000 \times 20{,}000$ (non sparse) matrix can eventually be considered. And in a close future, even more turbines with finer discretisations are expected.

Fig. 6 depicts a generic turbine layout. In that matter, a generic notation needs to be defined in order to synthetically describe the layout. Let us denominate by $A^{n_x \times n_y}_{xD,yD}$ a configuration with n_x rows consisting of n_y turbines each. Rows are



(a) Single turbine, mesh 15 × 23. (b) $A^{2\times 2}_{4D,1D}$, meshes 15 × 15.

Fig. 7. Representation of $|a_{ij}|$ for (left) a single turbine 15 × 23, corresponding to a matrix of size $N = 1{,}734$; and (right) $n = 4$ identical turbines with meshes 15 × 15 and layout $A^{2\times 2}_{4D,1D}$, corresponding to a matrix of size $N = 4{,}712$. The meshes are detailed in Table 1.

separated with a distance of xD (x diameters) from each other, and turbines in each row are separated with a distance of yD (y diameters) from each other. This given notation will be used as much as possible in the following.

3. Iterative solver approach

Owing to the unsteady frame of the computations, the influence system $A\boldsymbol{\mu} = \mathbf{b}$ described in Section 2.3 is solved at each time step. When a single turbine is considered, the influence matrix A is inverted and stored prior to the iterations, as mentioned earlier. And a single matrix-vector multiplication is then required at each time step. In the case of multiple turbine interactions [6], A is no longer constant with respect to time (see Fig. 5). Moreover, because there are several turbines, the size of the matrix A increases and matrix inversion at each time step becomes prohibitive in terms of CPU time consumption. For other numerical methods where the linear system is the core of the method, a very efficient solver needs to be used, using for example dedicated libraries. However, in the framework of particle methods with boundary integral methods (using Kutta–Joukowski emission, etc.), simple linear system resolutions are commonly used, although it can become very costly is some cases. In the following, an attempt to propose a fast and rigorous method is presented for unsteady problems with multi-bodies in the Lagrangian vortex particles framework.

3.1. Matrix characterisation

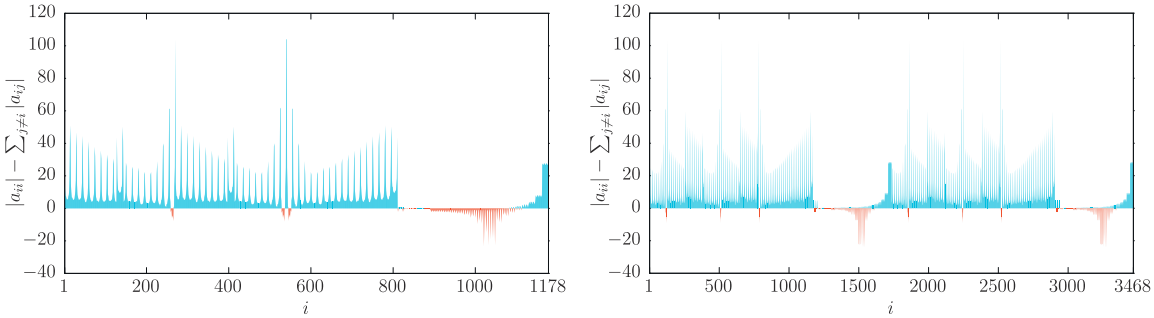
When attempting to choose the best numerical implementation, several questions arise such as: is the influence matrix sparse? Symmetric? Diagonally dominant? Positive-definite? From the definition of a_{ij} (Eqs. (17) and (18)) one can see that $a_{ij} \neq 0$ for all (i, j), except in some very hypothetical geometrical turbine design. As a consequence, the matrix A is generally not sparse.

Despite the formal symmetry of the original continuous problem, the matrix A is not symmetric either. As a matter of proof, a_{ij} depends on two geometrical items that are, the mesh element surface S_j (see Eq. (17) in a continuous form, and Eq. (18) in a discrete form) and the mesh element normal $\mathbf{n}_i$ (see Eq. (18)). Depending on the mesh element shape (during the mesh generation) and the regularity of these surface elements (if the quadrangle meshes are similar to squares of size dh^2), A might not be far from symmetry, as can be observed on Fig. 7. In order to quantify this asymmetry, let β represent the ratio of the norm of the anti-symmetric part of A (*i.e.* $1/2(A - A^T)$) to the norm of A:

$$\beta = \frac{\left\| (1/2)\left(A - A^T\right) \right\|}{\|A\|}. \tag{20}$$

One can see on Table 1 that β is rather small. This tends to corroborate that A is nearly symmetric. But when choosing the numerical implementation, the use of proper symmetric matrices cannot be argued.

Regarding the diagonal-dominance of A, owing to the gravitational-like interaction of Eq. (17), a diagonal dominance could have been expected. Nevertheless this mainly depends on the mesh element ordering during the mesh generation together with the relative distance between these elements. Even with a special care during the meshing phase, a real diagonal dominance cannot be achieved. As a matter of proof, two plots of $|a_{ii}| - \sum_{j\neq i} |a_{ij}|$ for different meshes and turbines configurations are depicted on Fig. 8. On Fig. 8a, the spiky parts correspond to the auto-influence of the blades on themselves, while the rest (on the right hand side) mainly corresponds to the influence with the hub surface. However, from



(a) Single turbine, mesh 15 × 15.

(b) $A_{2D}^{2\times 1}$, meshes 15 × 23.

Fig. 8. Plot of $|a_{ii}| - \sum_{j\neq i} |a_{ij}|$ with colouring of the sign for (left) a single turbine 15 × 15, corresponding to a matrix of size $N = 1{,}178$; and (right) $n = 2$ turbines with meshes 15 × 23, corresponding to a matrix of size $N = 3{,}468$. Blue means positive and red means negative. See Table 1 for mesh characteristics. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Number γ of negative eigenvalues of the matrix $(1/2)(A + A^T)$ for a configuration of $n = 2$ turbines modelled with different meshes for layout $A_{2D}^{2\times 1}$. The first three cases use the same mesh for both turbines, while the fourth case uses two different meshes.

Meshes	γ
$(5 \times 23, 5 \times 23)$	2
$(10 \times 5, 10 \times 5)$	0
$(15 \times 23, 15 \times 23)$	2
$(5 \times 23, 5 \times 15)$	2

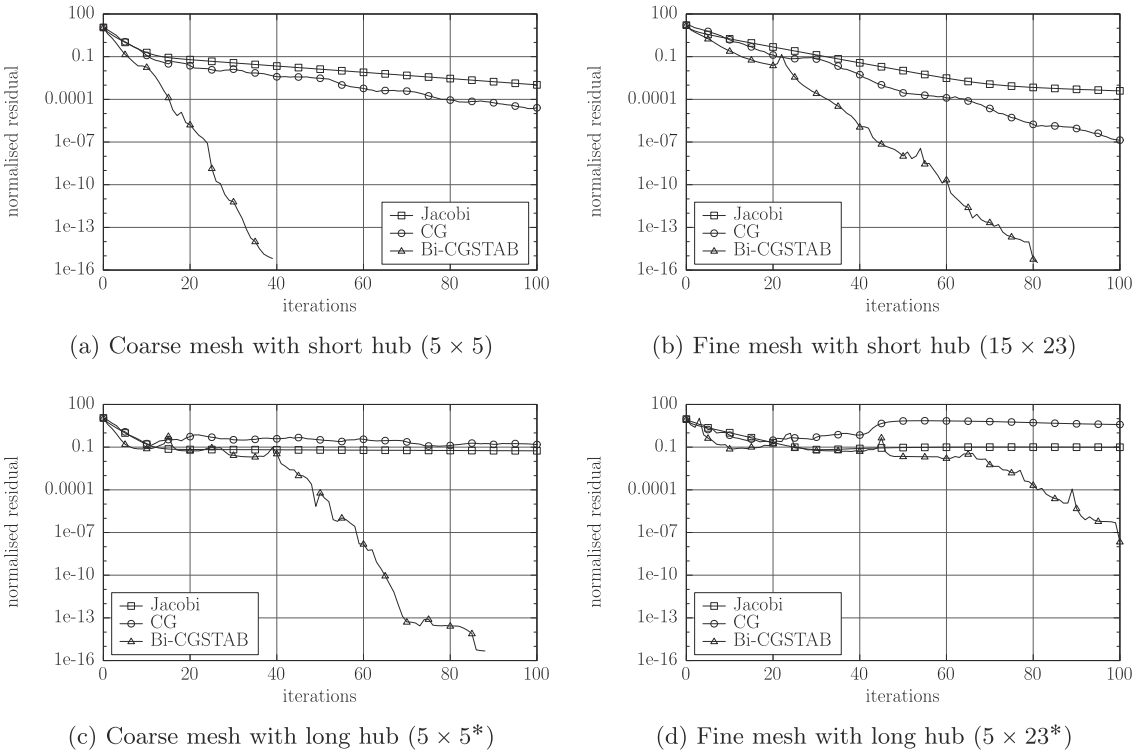
Fig. 8a, one can clearly observe that the matrix is not diagonally dominant. This property of non-diagonal dominance is not improved by adding more turbines in the configuration, as it actually tends to increase the weight of extra-diagonal terms. This statement is further corroborated by the results of Fig. 8b for a two-turbines configuration. Similar results (not presented here) were also obtained for larger numbers of turbines and different meshes.

The relative weakness of the influence terms between distinct turbines is visible by the fact that Fig. 8b looks like (but is indeed different from) periodic repetitions of a single pattern. Adding more turbines to the layout even emphasises this impression. Very short distances between turbines were chosen here in order to emphasise the influence of these extra-diagonal terms. Shorter distances between turbines could further increase the influence between turbines but may lead to non-physical turbine configurations. Indeed, for marine current turbine array configurations, the minimum lateral spacing is always of the order of one diameter (see [4,16,43]). The longitudinal spacing is generally of the order of magnitude of several diameters (between $4D$ to $6D$ for [4], minimum 10 diameters for [44], etc.) However, a qualitative observation of Fig. 7b tends to indicate that the matrices have a block-diagonal dominance. This observation will have a significant importance in the following of the paper (Section 3.3).

Finally, let us determine whether the matrix A is positive definite. A general non-symmetric matrix A is positive definite if its symmetric part (*i.e.* $(1/2)(A + A^T)$) is symmetric positive definite (SPD). In addition, a symmetric matrix is SPD if all its eigenvalues are positive. The number γ of negative eigenvalues of $(1/2)(A + A^T)$ is reported in Table 1 for single turbine configurations. It shows that the influence matrices are not always positive definite. In particular, the matrices $(1/2)(A + A^T)$ corresponding to the finer discretisations ($N_c \times 23$ and $5 \times 23^*$) have at least one negative eigenvalue (and at most two for $5 \times 23^*$). When several turbines configurations are considered, the number γ of negative eigenvalues, reported in Table 2, seems to be the sum of the number of negative eigenvalues of the separate single turbine case. One possible explanation could be the dominance of the diagonal blocks as compared to the other blocks of the matrix. Indeed it would follow that the matrix $(1/2)(A + A^T)$ has essentially the same spectrum as the matrix $(1/2)(K + K^T)$, where K is the block diagonal matrix whose blocks are the diagonal blocks of A, introduced later in Section 3.3 (see Eq. (22)).

3.2. Chosen implementation

From the previous section, it appears that iterative methods (in particular Krylov methods) would be the best suited to solve the influence system described in Section 2.3. A classical LU decomposition would require $\mathcal{O}(N^3)$ operations, which is unaffordable in terms of CPU time for large matrices. An iterative method is used as a way to reduce computational cost. Typically, an iterative method only involves a few matrix-vector multiplications per iteration, thus requiring $\mathcal{O}(mN^2)$



(a) Coarse mesh with short hub (5×5)

(b) Fine mesh with short hub (15×23)

(c) Coarse mesh with long hub ($5 \times 5^*$)

(d) Fine mesh with long hub ($5 \times 23^*$)

Fig. 9. Comparison of the convergence of the normalised residual $\tilde{r}^{(i)} = \|A\boldsymbol{\mu}^{(i)} - \mathbf{b}\|/\|\mathbf{b}\|$ as a function of the iterations for the Jacobi, the Conjugate Gradient (CG) and the Bi-CGSTAB methods with four different turbine meshes (see Table 1).

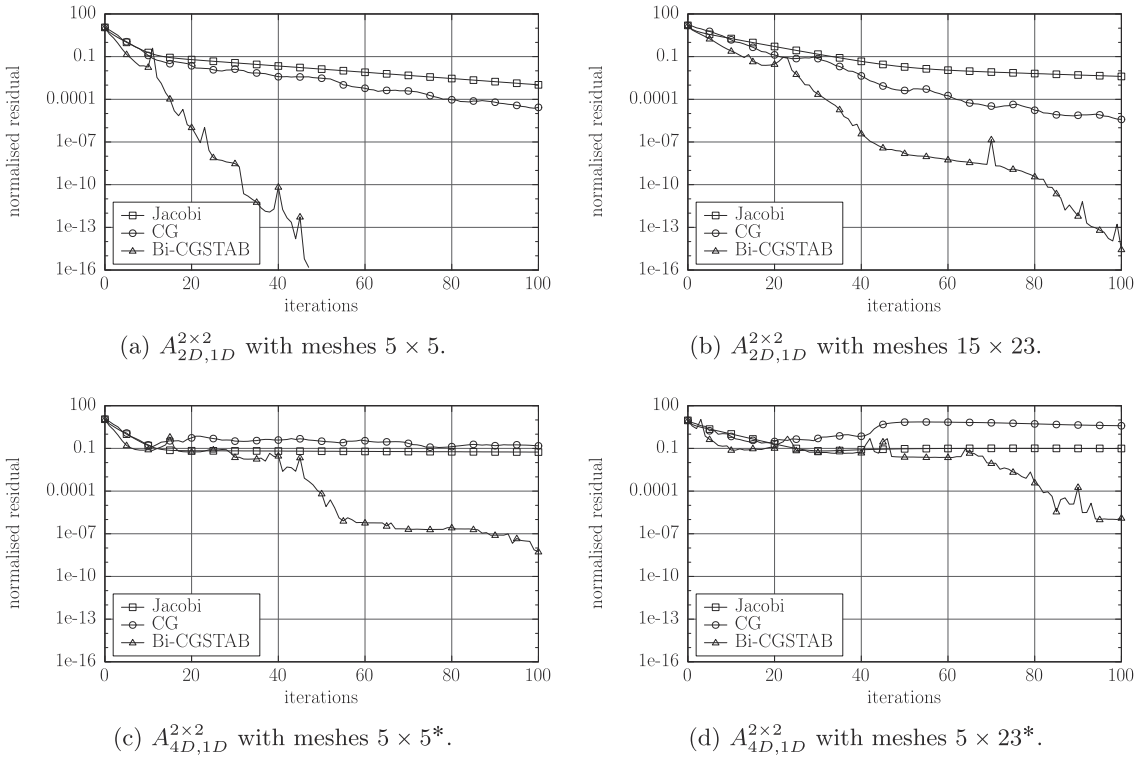
operations for a total of m iterations. An iterative method is thus more interesting than a direct solver provided that it converges (to a desired level of accuracy) in less than N iterations. This is generally the case of Krylov methods.

The chosen method is the Bi-Conjugate Gradient Stabilised (abbreviated Bi-CGSTAB) described in van der Vorst [22]. The Bi-CGSTAB algorithm is a Krylov method adapted to general linear systems. This algorithm can be seen as an improvement of other general methods such as the Conjugate Gradients-Squared method [22]. Because the Conjugate Gradient (CG) is designed for symmetric matrices, this method might not be the best choice here. However, as the considered matrices are only slightly non-symmetric, a CG implementation is also tested in the sequel. Finally, a classical Jacobi implementation is also considered for general comparison. The Jacobi method is a classical iterative method for solving linear systems. It is simple to implement and to parallelize. However, its convergence is not guaranteed for matrices that are not diagonally dominant, and may be very slow.

It should be noted at this point that for Krylov methods (such as the Bi-CGSTAB or the CG algorithms mentioned above), one usually needs to compute matrix-vector products having a computational complexity of $\mathcal{O}(N^2)$. For Laplace equations in integral form like the one considered here (see Eq. (8)), such matrix-vector products can be efficiently computed using the *Fast Multipole Method* (FMM) [45–48], leading to a theoretical complexity of $\mathcal{O}(N)$ for the products, and thus a total complexity of $\mathcal{O}(mN)$ for the iterative resolution in m iterations. Although the constant in the $\mathcal{O}(N)$ complexity is quite large, FMM can still provide substantial acceleration for large systems [47]. Because the details, in terms of error criterion and parallelization strategy, are rather involved, and because the computations presented hereafter are still relatively small in terms of matrix size (a few thousand degrees of freedom), this acceleration technique is not considered in this paper. Nevertheless, the discussion and results presented in this paper would still remain valid using FMM acceleration techniques, which would indeed make the solving even more efficient. The reader may refer to [48] for a description and implementation details of FMM in the context of panel methods.

Fig. 9 shows a comparison of the convergence of the three implemented algorithms, in the case of a single turbine. Convergence is characterised by the normalised residual $\tilde{r}^{(i)} = \|A\boldsymbol{\mu}^{(i)} - \mathbf{b}\|/\|\mathbf{b}\|$, where A represents the influence matrix of the considered configuration, $\mathbf{b}$ the right hand side of the influence system, and $\boldsymbol{\mu}^{(i)}$ the approximation of the solution $\boldsymbol{\mu}$ given by the ith iteration of the algorithm. The Jacobi method seems to converge quite slowly and after 100 iterations, a normalised residual of 10^{-3} is hardly achieved. The CG method converges a little faster. The Bi-CGSTAB is the only method to reach machine precision in less than 100 iterations in the single turbine case.

Fig. 10 shows convergence plots for 4-turbine configurations. Similarly to the single turbine case, the Bi-CGSTAB behaves better than the CG and Jacobi methods. For instance, in the configuration depicted on Fig. 10a, the Bi-CGSTAB reaches the same precision ($\tilde{r}^{(i)} \approx 10^{-4}$) in less than 20 iterations as the CG does in 80 iterations. Furthermore, the CG does not converge



(a) $A^{2\times 2}_{2D,1D}$ with meshes 5×5.

(b) $A^{2\times 2}_{2D,1D}$ with meshes 15×23.

(c) $A^{2\times 2}_{4D,1D}$ with meshes $5 \times 5^*$.

(d) $A^{2\times 2}_{4D,1D}$ with meshes $5 \times 23^*$.

Fig. 10. Comparison of the convergence of the normalised residual for the Jacobi, the Conjugate Gradient (CG) and the Bi-CGSTAB methods for two different layouts of 4 turbines (see Table 1).

for the last two configurations (Fig. 10c and d). As mentioned previously, this is most likely due to the fact that the CG is not suited to non-symmetric matrices. Its use is thus unsafe as the directions are not properly conjugate (specifically, the metric used for conjugation is not a proper inner product), which means that the resulting Krylov subspace is erroneous. For nearly symmetric matrices, the erroneous directions might still be good enough to keep the solution improving iteratively. On the other hand, some matrices may drive the solution outside the theoretical Krylov space, potentially leading to situations from which the CG iterations cannot recover. At any rate, even considering the fact that the Bi-CGSTAB requires twice as much computation (in terms of matrix-vector products) as the CG, the Bi-CGSTAB still appears to be the most efficient method in solving this problem.

3.3. Preconditioning

It is well-known that an adequate preconditioner may dramatically improve the convergence of such iterative methods. The purpose of preconditioning is basically to improve the condition number of the matrix of the linear system. In the sequel, we will focus on left preconditioning, which consists, for a linear system of the form $Ax = b$, in solving the equivalent system $M^{-1}Ax = M^{-1}b$ where $M^{-1}A$, the preconditioned matrix, has a smaller condition number than A. The matrix M (or sometimes M^{-1} itself) is then called the preconditioner. One first simple choice of preconditioner would be the Jacobi preconditioner, which consists in taking the diagonal of A as the preconditioner M. Let us denote it by D. It has the following definition:

$$D = (d_{ij}) = \mathrm{diag}(A), \qquad d_{ij} = a_{ii}\delta_{ij}, \quad \forall i, j = 1, \ldots, N, \tag{21}$$

where δ_{ij} denotes the Kronecker delta. The advantage of such a preconditioner is that its inverse is easy to compute. As a matter of fact, D^{-1} is simply the diagonal matrix whose elements are the inverses of those of D.

Taking advantage of the structure and evolution of the matrix (with respect to the time-steps of the unsteady simulation), let us now define a more appropriate preconditioner. First, recall that the matrix of the influence equation has a natural block structure (see Eq. (19)) and that its diagonal blocks have essentially larger coefficients than the others (see Fig. 7). Moreover, the diagonal blocks of the matrix are constant over time. This fact allows the diagonal blocks to be used for preconditioning. Let K be the block diagonal matrix whose blocks correspond to the diagonal blocks of the matrix A. The matrix K is then used as the preconditioning matrix. Its advantage is that its inverse is also a block diagonal matrix

Table 3
Number of iterations before convergence for the different algorithms using layout $A^{2\times 2}_{2D,1D}$ and different mesh sizes (with short hubs). The values in the entries represents the first number of iterations i for which the relative residual $\tilde{r}^{(i)} = \|A\boldsymbol{\mu}^{(i)} - \mathbf{b}\|/\|\mathbf{b}\|$ reaches machine precision $\epsilon \approx 2.2 \cdot 10^{-16}$. When the algorithm did not reach the required precision before 500 iterations, the order of magnitude of the final normalised residual is given between parentheses.

	No preconditioning			Jac. precond.		BJac. precond.	
Mesh	Jacobi	CG	Bi-CGSTAB	CG	Bi-CGSTAB	CG	Bi-CGSTAB
5×5	(10^{-12})	(10^{-10})	47	(10^{-3})	33	5	3
5×7	(10^{-12})	329	53	(10^{-5})	34	5	3
5×11	(10^{-11})	(10^{-12})	52	(10^{-7})	33	5	3
5×15	(10^{-10})	(10^{-4})	67	(10^{-1})	39	5	3
5×23	(10^{-8})	(10^{-5})	89	(10^{1})	37	5	3
10×5	(10^{-9})	(10^{-6})	76	(10^{-2})	41	5	3
10×7	(10^{-9})	(10^{-11})	89	(10^{-8})	42	5	3
10×11	(10^{-9})	219	88	(10^{-8})	42	5	3
10×15	(10^{-8})	373	83	(10^{-3})	43	5	3
10×23	(10^{-7})	(10^{-12})	92	(10^{0})	43	5	3
15×5	(10^{-8})	(10^{-6})	96	(10^{-3})	49	5	3
15×7	(10^{-8})	(10^{-12})	101	(10^{-4})	52	5	3
15×11	(10^{-8})	218	101	(10^{-10})	49	5	3
15×15	(10^{-8})	224	109	(10^{-7})	47	5	3
15×23	(10^{-7})	471	103	(10^{-4})	50	5	3

Table 4
Number of iterations before convergence for the considered algorithms using layout $A^{2\times 2}_{4D,1D}$ and different mesh sizes (with long hubs). See also Table 3.

	No preconditioning			Jac. precond.		BJac. precond.	
Mesh	Jacobi	CG	Bi-CGSTAB	CG	Bi-CGSTAB	CG	Bi-CGSTAB
$5 \times 5^*$	(10^{-2})	(10^{-2})	194	(10^{0})	94	5	3
$5 \times 7^*$	(10^{-2})	(10^{-2})	193	(10^{-2})	123	5	3
$5 \times 11^*$	(10^{-1})	(10^{1})	269	(10^{1})	150	5	3
$5 \times 15^*$	(10^{-1})	(10^{1})	301	(10^{0})	164	5	3
$5 \times 23^*$	(10^{-1})	(10^{0})	288	(10^{9})	143	5	3

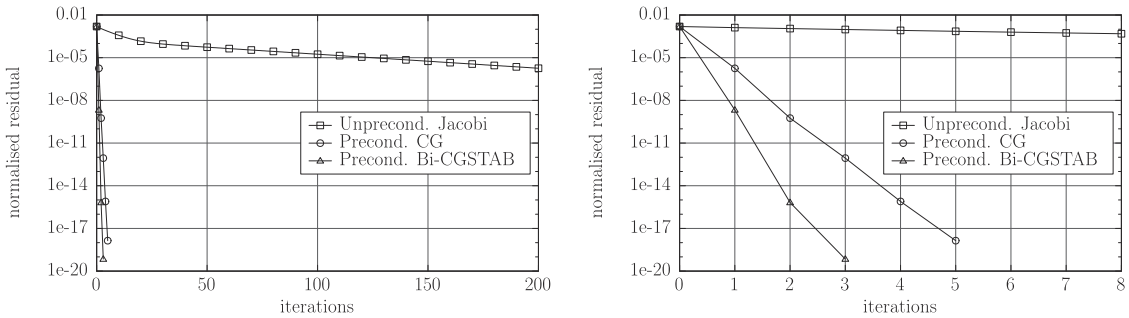
consisting of the inverses of the blocks of A. More precisely, K and its inverse K^{-1} have the following form:

$$K = \begin{bmatrix} [A_{11}] & 0 & \cdots & 0 \\ 0 & [A_{22}] & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & [A_{nn}] \end{bmatrix}, \quad K^{-1} = \begin{bmatrix} [A_{11}]^{-1} & 0 & \cdots & 0 \\ 0 & [A_{22}]^{-1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & [A_{nn}]^{-1} \end{bmatrix}. \tag{22}$$

Since the matrix K is constant over time, its inversion can be performed at the beginning of the simulation. This can be done by using a direct method. Indeed, the diagonal blocks of the matrix A are invertible since they correspond to the well defined problem of the influence of a single turbine on itself. In what follows, the matrix K will be called the block Jacobi (abbreviated BJac.) preconditioner.

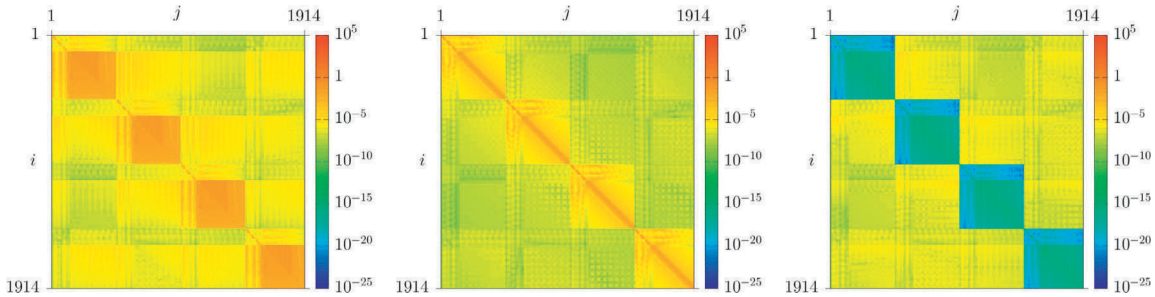
Tables 3 and 4 show examples of the number of iterations for unpreconditioned and preconditioned (with the Jacobi (Jac.) and the block Jacobi (BJac.) preconditioners) for both the CG and the Bi-CGSTAB. Let us first make a few comments on the simple Jacobi preconditioner mentioned previously. When combined with the CG method, the Jacobi preconditioner actually makes the iterations worse (degrading as compared to the unpreconditioned version). This combination of the Jacobi preconditioner with the CG is thus inefficient here. However, the Jacobi preconditioner combined with the Bi-CGSTAB always improves the convergence (see Tables 3 and 4). As a matter of fact, even in the worse cases of Table 4, approximately half as many iterations are required to reach machine precision with the use of the Jacobi preconditioner (as compared to the unpreconditioned version).

Second, using the block Jacobi preconditioner, the improvement appears clearly as the number of iterations decreases dramatically for the CG and Bi-CGSTAB algorithms. For all of the cases considered, the number of iterations necessary to obtain a relative residual $\tilde{r}^{(i)}$ using the preconditioned Bi-CGSTAB below the machine precision is 3. The CG method does not reach the required precision within a reasonable number of iterations for several unpreconditioned computations. At the same time, it converges in 5 iterations when preconditioned in a block Jacobi manner. Furthermore, with this preconditioning, the number of iterations does not change when the mesh structure is modified.



(a) 200 iterations (b) First eight iterations

Fig. 11. Comparison of the convergence of the normalised residual $\tilde{r}^{(i)}$ for the unpreconditioned Jacobi method, the BJac. preconditioned CG and the BJac. preconditioned Bi-CGSTAB. The configuration corresponds to an aligned $A^{2\times 2}_{2D,1D}$ layout with meshes 15×23.



(a) Matrix A. Unprecond. (b) Matrix $D^{-1}A$. Jac. precond. (c) Matrix $K^{-1}A$. BJac. precond.

Fig. 12. Comparison of the structures of the non preconditioned and preconditioned influence matrix for the 4 meshes $5 \times 5^*$ in configuration $A^{2\times 2}_{4D,1D}$ of Table 4.

To further illustrate the effectiveness of this BJac. preconditioner, Fig. 11 shows the convergence of the BJac. preconditioned methods (CG and Bi-CGSTAB) compared to the basic unpreconditioned Jacobi iterative method. After 200 iterations, the latter still has a residual greater than 10^{-6}, while the preconditioned CG and BI-CGSTAB get below 10^{-6} after 2 and 1 iteration(s), respectively. It should be mentioned that the CG is slightly faster than the Bi-CGSTAB due to the fact that the latter requires twice as many matrix-vector multiplications at each iteration. However it has to be recalled that the use of the CG is still not safe here due to the general asymmetry of the matrix A (see Section 3.2).

To conclude, an example of the effect of the preconditioners on the matrix A is shown on Fig. 12. In particular, one can see that the Jacobi preconditioner leaves the matrix with many large values outside the diagonal.

4. Application: computations of several turbines in interaction

The following paragraphs present some results as an illustrative application of the iterative solver approach described above. Here, two configurations of elementary interactions between marine current turbines in close proximity are presented. Marine current turbines are considered here because it corresponds to the background research topics of our research team. However, applications to wind turbines could be considered without any restriction. The results presented here correspond to computations that were recently made accessible thanks to CPU time improvements brought by the iterative solver approach. A complete numerical study is to be performed in the near future, including the influence of the distance between turbines, as well as other numerical and physical parameters. Although qualitative results of wake interaction in a three-turbine setting are briefly presented, the main focus of this section will remain on the quantitative assessment of the savings achieved in terms of computational time thanks to the iterative solver approach presented in the previous sections.

4.1. Wake interaction between three turbines

The two configurations considered hereafter are illustrated as the green and red boxes in Fig. 13. The first configuration (green in Fig. 13) consists of two rows of turbines, and has already been studied experimentally by, among others, Kervella et al. [49]. The second configuration (red in Fig. 13) involves 10 turbines and is basically a repetition of the triangular pattern of the first configuration.

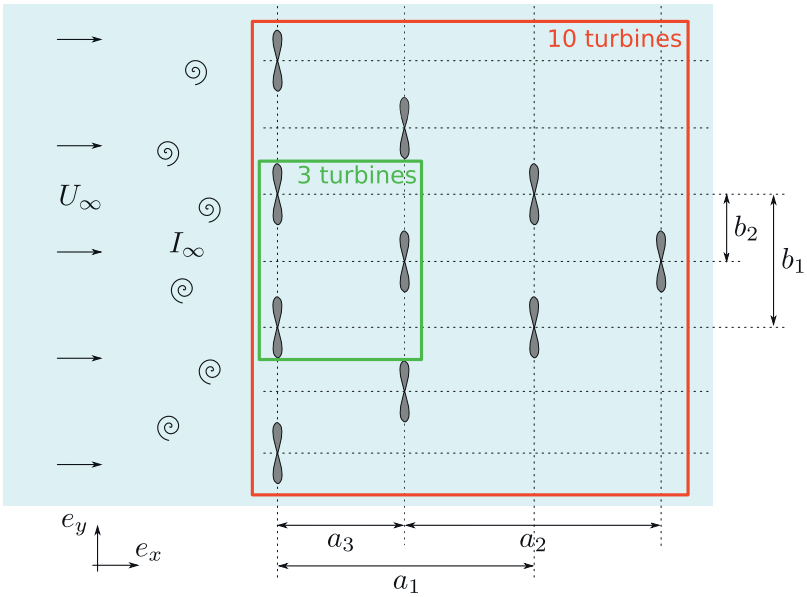


Fig. 13. Marine current turbines array with examples of elementary interactions.

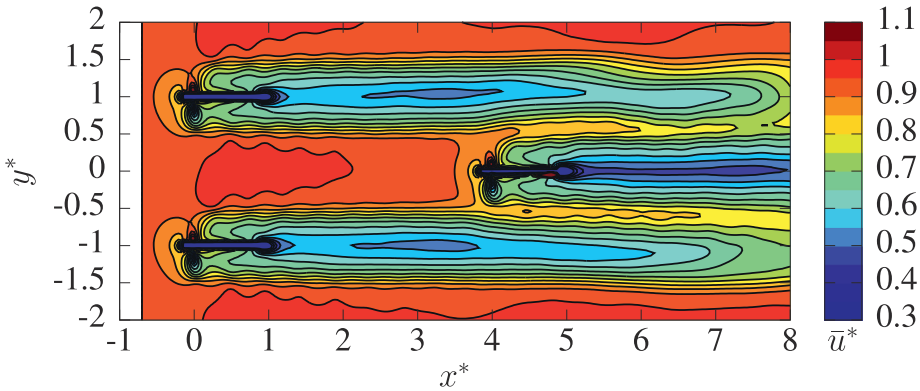


Fig. 14. Numerical wake for the 3-turbine configuration (green in Fig. 13) at TSR = 2. The inter-device distances are $a_3 = 4D$ and $b_1 = 2b_2 = 1D$ using the notations from Fig. 13. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 14 depicts the instantaneous velocity field in the wake of three turbines, with a layout inspired from the experimental work of Kervella et al. [49] and illustrated by the green box in Fig. 13. Using the notations from Fig. 13, the inter-device distances considered here correspond to $a_3 = 4D$ and $b_1 = 2b_2 = 2D$. The instantaneous velocity field was obtained at $t \approx 28.2\,s$ (physical time) in the unsteady computation, corresponding to 399 unsteady iterations of time step $dt = 0.07066\,s$, using three meshes $5 \times 5^*$ from Table 1. The reader may refer to [5] for a discussion on how the time step is chosen. This computation corresponds to the coarsest mesh discretisation, and improvements in the quality of the results are expected in the near future. However, these results are still interesting qualitatively as many physical aspects are already observable even for these discretisations, such as the wake interaction that is clearly visible in Fig. 14 where the wake of the first row of turbines reaches the third turbine. As already discussed, a complete survey including several turbine layouts, several inter-device distances and other physical parameters is scheduled in the near future. Comparison with the experimental work of Kervella et al. [49] will also be considered.

4.2. Assessment of CPU time savings using the present approach

In order to have an assessment of the CPU time savings obtained with the present approach, the last five meshes of Table 1, corresponding to the longer hub with $N_c^{\text{hub}} = 58$, were tested on the 3-turbine and 10-turbine configurations depicted in Fig. 13. The 3-turbine configuration corresponds exactly to the one presented in Fig. 14. These meshes were chosen for two reasons: first because the longer hub corresponds to the real size of the turbine hub used in the experiments (see [5,42]), and second because the mesh $5 \times 23^*$ also corresponds to the larger number of mesh elements. In the case of 10

Table 5
Total emission time t_{tot} (in seconds) and its decomposition for each mesh of Table 5. Both the developed preconditioned Bi-CGSTAB and the Direct approaches are presented for the 3-turbine and the 10-turbine configurations.

Mesh	Precond. Bi-CGSTAB				Inversion			
3 turb.	t_1	t_2	t_3	t_{tot}	t_1	t_2	t_3	t_{tot}
$5 \times 5^*$	3.2×10^{-3}	3.1×10^{-3}	3.0×10^{-3}	9.4×10^{-3}	4.4×10^{-3}	4.6×10^{-2}	3.2×10^{-3}	5.3×10^{-2}
$5 \times 7^*$	5.2×10^{-3}	3.1×10^{-3}	6.2×10^{-3}	1.4×10^{-2}	8.1×10^{-3}	9.4×10^{-2}	6.8×10^{-3}	1.1×10^{-1}
$5 \times 11^*$	1.3×10^{-2}	3.7×10^{-3}	1.6×10^{-2}	3.2×10^{-2}	1.9×10^{-2}	2.8×10^{-1}	1.6×10^{-2}	3.1×10^{-1}
$5 \times 15^*$	2.2×10^{-2}	3.8×10^{-3}	2.6×10^{-2}	5.2×10^{-2}	3.0×10^{-2}	5.8×10^{-1}	2.6×10^{-2}	6.3×10^{-1}
$5 \times 23^*$	4.8×10^{-2}	4.8×10^{-3}	6.0×10^{-2}	1.1×10^{-1}	6.3×10^{-2}	$3.5 \times 10^{+0}$	6.0×10^{-2}	$3.6 \times 10^{+0}$
10 turb.	t_1	t_2	t_3	t_{tot}	t_1	t_2	t_3	t_{tot}
$5 \times 5^*$	4.4×10^{-2}	4.8×10^{-3}	4.4×10^{-2}	9.3×10^{-2}	4.7×10^{-2}	$1.1 \times 10^{+0}$	4.5×10^{-2}	$1.2 \times 10^{+0}$
$5 \times 7^*$	7.2×10^{-2}	7.0×10^{-3}	7.4×10^{-2}	1.5×10^{-1}	7.8×10^{-2}	$3.0 \times 10^{+0}$	7.5×10^{-2}	$3.2 \times 10^{+0}$
$5 \times 11^*$	1.5×10^{-1}	3.5×10^{-2}	1.6×10^{-1}	3.5×10^{-1}	1.7×10^{-1}	$1.7 \times 10^{+1}$	1.6×10^{-1}	$1.7 \times 10^{+1}$
$5 \times 15^*$	2.6×10^{-1}	5.1×10^{-2}	2.8×10^{-1}	5.9×10^{-1}	3.4×10^{-1}	$4.4 \times 10^{+1}$	2.8×10^{-1}	$4.4 \times 10^{+1}$
$5 \times 23^*$	5.5×10^{-1}	9.0×10^{-2}	6.3×10^{-1}	$1.3 \times 10^{+0}$	6.1×10^{-1}	$1.5 \times 10^{+2}$	6.3×10^{-1}	$1.5 \times 10^{+2}$

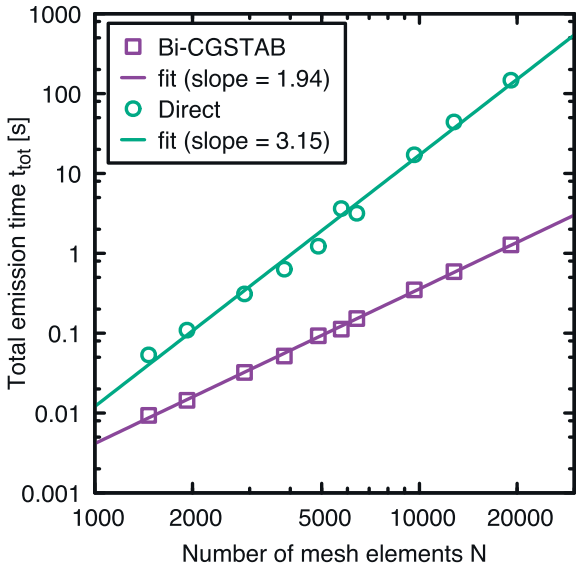


Fig. 15. Total emission time t_{tot} as a function of the number of mesh elements N for both the preconditioned Bi-CGSTAB and the direct approaches. The 3-turbine and 10-turbine configurations illustrated in Fig. 13 were tested for all meshes of Table 5.

turbines in interaction, this last mesh will be considered as a critical case with a 19, 140 $\times$ 19, 140 matrix system to be solved at each time step either by direct inversion or by iterative solve using the preconditioned Bi-CGSTAB method.

Table 5 presents the particle emission time for both configuration (with 3 and 10 turbines) and different meshes. This total emission time t_{tot} includes the time t_1 spent by the matrix update (Eq. (18)) and the emitted particle initialisation, the time t_2 spent either by the system resolution in the case of the use of Bi-CGSTAB or by the matrix inversion and matrix-vector multiplication for the direct method case, and the time t_3 spent by the computation of the right-hand-side (Eq. (16)). These times were obtained from the first 10 iterations where an average was performed over the last 9 values, the first iteration being discarded. An average over the first 100 iterations was also performed without significant changes in the resulting values. Except for the computation of the right-hand-side, where the time t_3 increases with the number of emitted particles, the other times remain nearly identical regardless of the global iteration in the unsteady computation.

Fig. 15 depicts the particle emission time t_{tot} as a function of the total number of mesh elements (from $N = 1{,}464$ to $N = 19{,}140$), leading to an influence matrix of size $N \times N$. One can observe that the preconditioned Bi-CGSTAB implementation is much faster than the direct case for all configurations. Using linear regression in a log-log scale, a slope of 1.94 was found for the Bi-CGSTAB implementation with a correlation coefficient of 0.99. Similarly, a slope of 3.15 (also with correlation coefficient of 0.99) was obtained for the matrix inversion, which corresponds to the theoretical order of $\mathcal{O}(N^3)$ for direct solvers. These results show that the matrix inversion implementation is getting prohibitively expensive as the number of degrees of freedom increase, while the preconditioned Bi-CGSTAB is always much faster for the cases considered and has a cost that increases more slowly with N. In the most extreme case presented here (for 10 turbines with the $5 \times 23^*$ mesh), a difference of more than 2 orders of magnitude is obtained.

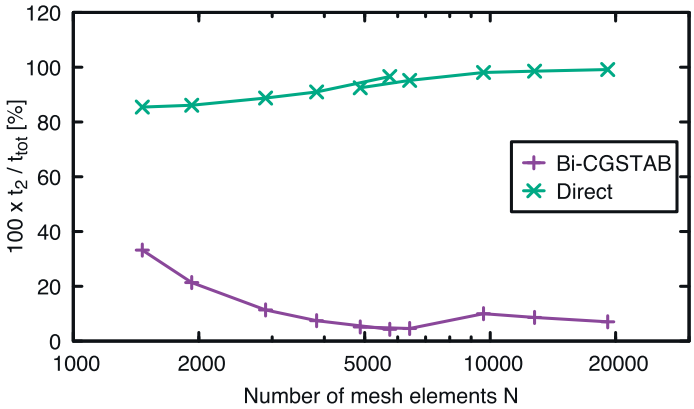


Fig. 16. Proportion of t_2 as a percentage of t_{tot} for the all meshes of Table 5 for the 3-turbine and the 10-turbine configurations, for both the preconditioned Bi-CGSTAB and the direct solver.

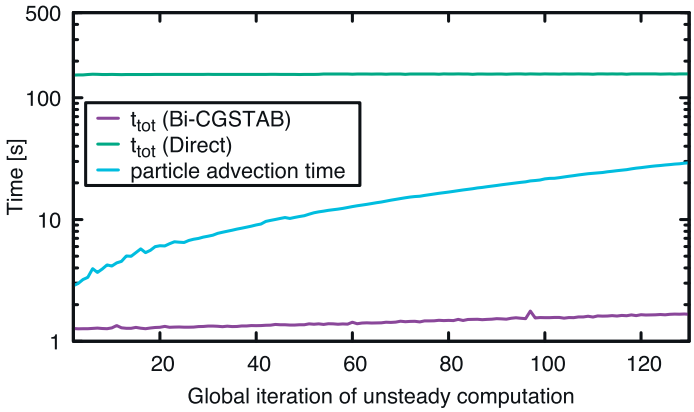


Fig. 17. Emission time t_{tot} compared with advection time for the first 130 iterations of a 10-turbine configuration using $5 \times 23^*$ meshes.

Fig. 16 depicts the proportion of time t_2 of the matrix system resolution (either using the preconditioned Bi-CGSTAB or the direct inversion) as a percentage of the total emission time t_{tot} for all the considered meshes. It illustrates that the reason why the total emission time t_{tot} fits well with the $\mathcal{O}(N^3)$ order for the direct case is basically because the inversion time t_2 represents more than 85% of the total emission time t_{tot} for all the configurations (*i.e.* matrix sizes) considered. As expected, the total emission time is dominated by the matrix inversion, which prohibits its use such configurations. For the finer 3 meshes in the 10-turbine configuration, t_2 represents even more that 98%. On the contrary, for the preconditioned Bi-CGSTAB implementation, t_2 represents less than 37% for all configurations and falls down to less than 10% for the finer discretisations with the 10-turbine configuration. This means that the slope of 1.94, corresponding approximately to a $\mathcal{O}(N^2)$ trend for the particle emission is primarily due to the other parts of the emission scheme (*i.e.* the matrix update, particle initialisation and right-hand-side computation), rather than the iterative solve itself. This result is promising as additional CPU time savings may eventually be made if the other parts of the emission procedure can be further optimised in terms of computational efficiency.

It should be recalled that the iterative solve is also supposed to have an $\mathcal{O}(N^2)$ computational complexity, more specifically $\mathcal{O}(mN^2)$, where m represents the number of Bi-CGSTAB iterations. As illustrated in Section 3, because the block Jacobi preconditioner is particularly adapted to the multi-turbine case, the number of Bi-CGSTAB iterations m is expected to be small. We actually checked that this number remains small throughout the computations, and observed that it usually remained below 10 at the beginning of the computations, and then stabilizes around a value below 5 at the end of the computations.

In the Lagrangian vortex framework, it is well know that the particle advection is critical in terms of computational time and, at the end of computations, when the number of particles becomes very large as compared to the constant number of mesh elements, the emission procedure is expected to represent only a minor fraction of total CPU time within a given unsteady iteration. This is generally true and this was the case for a single turbine configuration. However, this is generally no longer the case for the 10-turbine configurations presented here. To illustrate this, Fig. 17 depicts the total emission time t_{tot} for both the pre-conditioned Bi-CGSTAB and the direct solver compared to the advection time for the first 130 iterations of the 10-turbine configuration using $5 \times 23^*$ meshes. On this figure, it can be observed that the advection time is increasing

with the number of iterations (or equivalently the physical time), which was expected because of the increasing number of emitted particles, and hence of total particles, as the computation progresses. Additionally, one can observe that the total emission time for the direct approach never becomes lower than the advection time, even after 130 iterations where 120, 832 particles are involved. At this stage, it is still the beginning of the wake interaction computation but it can be easily understood that the considered emission time will never become negligible and, although it may, after a large number of iterations, become smaller than the advection time, it will still represent a large amount of CPU time per iteration. On the contrary, with the proposed preconditioned Bi-CGSTAB implementation, the total emission time t_{tot} always remains lower than the advection time, and rapidly becomes negligible with nearly an order of magnitude after a 100 iterations. This last Fig. 17 really confirms the need of such an implementation in order to undertake more accurate computations of several turbines in interaction.

5. Conclusion

The paper presents the recent numerical development carried out on unsteady 3D software [5] developed at LOMC in partnership with IFREMER. These developments were necessary in the case of several turbines (marine or wind turbines) in order to reduce the computational cost of such simulations. Indeed, a linear system needs to be solved at each time step in order to enforce the boundary conditions on the turbine blades. When a single turbine is considered, the so-called influence matrix is constant over time and can be inverted at the beginning of the computation and the inverse is stored. Solving the linear system then only consists of a single matrix-vector multiplication. Accurate computations can be performed in such single turbine configurations as presented in Pinon et al. [5]. However, when several turbines are considered, the influence matrix is not constant. Mycek et al. [6] computed a two turbine configuration by the use of a direct matrix inversion at each time step. This direct matrix inversion technique was computationally expensive and prohibits either the use of finer mesh discretisations or more complex turbines layouts.

An iterative approach was then decided in order to solve the linear system. Firstly, a precise characterisation of the influence matrix was necessary in order to chose the best implementation. The influence matrix is, by construction, neither sparse nor symmetric, despite the symmetry of the continuous formulation. This is due to the mesh itself (element surfaces and normals) which prevent the discrete influences from being symmetric. However, the defined matrices can be qualified as “close” to symmetry. Then, the matrices proved not to be diagonally dominant and some of them not positive definite. Regarding the positive definition, some matrices finally showed to have a symmetric part with only a couple of negative eigenvalues preventing from a general positive definite property.

Despite these properties, three iterative methods were tested. The Jacobi, the CG and the Bi-CGSTAB methods were implemented keeping in mind that the Jacobi method is known to have a very slow convergence and that the CG is designed for symmetric matrices. The Jacobi and CG were used as a matter of comparison. Several turbine configurations were tested. In order to emphasise the importance of the matrix elements representing the interaction between turbines (extra-diagonal blocks), configurations with small inter-distances were preferred. This does not exactly fits with real wind or marine current turbines arrays configuration, although shorter inter-device distances are expected for marine application. Of course, the results are applicable for longer inter-device distances.

After some numerical tests, the Bi-CGSTAB method proved to be the more efficient one where the simple Jacobi and the CG methods were failing for some configurations. Whatever the turbines configuration, the Bi-CGSTAB method always obtained convergence with a normalised residual close to machine precision after about a hundred iterations at most. However, both the Jacobi and the CG methods hardly never reached a normalised residual lower than 10^{-3} after the same number of iterations. And in most cases, they never achieved machine precision after 500 iterations.

As a matter of further improvement, a preconditioner was tested. A simple Jacobi pre-conditioner was tested without much success. But the application of a block-Jacobi preconditioner always gave accurate results (below machine precision) after 5 iterations for the CG and 3 iterations for the Bi-CGSTAB methods. It should be stressed that 5 iterations of the preconditioned CG is nearly as computationally expensive as 3 iterations of the preconditioned Bi-CGSTAB. Nevertheless, because the CG is not suited to non-symmetric matrices, the choice of the block-Jacobi preconditioned Bi-CGSTAB algorithm was finally made.

Regarding application, the present study mainly focuses on numerical computations of interactions in a marine current turbine farm. Preliminary results on staggered configurations involving 3 turbines, similarly to the configuration of Kervella et al. [49], as well as bigger cases with 10 turbines, are presented in terms of instantaneous wake velocity maps. The CPU time measurements clearly demonstrate the improvements brought by the use of the preconditioned iterative solver instead of the direct Gauss–Jordan inversion method previously used. These test cases give confidence in the numerical tool, and show that computations of arrays with several turbines are closer to being accessible at reasonable computational cost.

Acknowledgement

The authors would like to thank Haute-Normandie Regional Council and Institut Carnot IFREMER Edrome for their financial supports for C. Carlier’s Ph.D. grant and “GRR Energie” programs. The authors also would like to thank the CRIANN (Centre Régional Informatique et d’Applications Numériques de Normandie) for their available numerical computation resources.

References

[1] M. Hansen, J. Sørensen, S. Voutsinas, N. Sørensen, H. Madsen, State of the art in wind turbine aerodynamics and aeroelasticity, Prog. Aerosp. Sci. 42 (4) (2006) 285–330, doi:10.1016/j.paerosci.2006.10.002. URL http://www.sciencedirect.com/science/article/pii/S0376042106000649.

[2] A. Miller, B. Chang, R. Issa, G. Chen, Review of computer-aided numerical simulation in wind energy, Renew. Sustain. Energy Rev. 25 (0) (2013) 122–134, doi:10.1016/j.rser.2013.03.059. URL http://www.sciencedirect.com/science/article/pii/S1364032113002189.

[3] K.-W. Ng, W.-H. Lam, K.-C. Ng, 2002–2012: 10 years of research progress in horizontal-axis marine current turbines, Energies 6 (3) (2013) 1497–1526, doi:10.3390/en6031497. URL http://www.mdpi.com/1996-1073/6/3/1497.

[4] M.J. Churchfield, Y. Li, P.J. Moriarty, A large-eddy simulation study of wake propagation and power production in an array of tidal-current turbines, Philos. Trans. R. Soc. Lond. A: Math., Phys. Eng. Sci. 371 (1985) (2013), doi:10.1098/rsta.2012.0421. ref. 20120421, arXiv: http://rsta.royalsocietypublishing.org/content/371/1985/20120421.full.pdf.

[5] G. Pinon, P. Mycek, G. Germain, E. Rivoalen, Numerical simulation of the wake of marine current turbines with a particle method, Renew. Energy 46 (0) (2012) 111–126, doi:10.1016/j.renene.2012.03.037. URL http://www.sciencedirect.com/science/article/pii/S0960148112002418.

[6] P. Mycek, B. Gaurier, G. Germain, G. Pinon, E. Rivoalen, Numerical and experimental study of the interaction between two marine current turbines, Int. J. Marine Energy 1 (0) (2013) 70–83, doi:10.1016/j.ijome.2013.05.007. URL http://www.sciencedirect.com/science/article/pii/S2214166913000088.

[7] J. Baltazar, J.A.C. Falcão de Campos, Hydrodynamic analysis of a horizontal axis marine current turbine with a boundary element method, in: Proceedings of the ASME 27th Conference on Offshore Mechanics and Arctic Engineering (OMAE), ASME, 2008, pp. 883–893, doi:10.1115/OMAE2008-58033. http://link.aip.org/link/abstract/ASMECP/v2008/i48234/p883/s1. Estoril, Portugal

[8] T. McCombes, C. Johnstone, A. Grant, Unsteady wake modelling for tidal current turbines, IET Renew. Power Gener. 5 (2011) 299–310. (11), URL http://digital-library.theiet.org/content/journals/10.1049/iet-rpg.2009.0203.

[9] B. Marichal, F. Hauville, Numerical calculation af an incompressible, inviscid three-dimensional flow about a wind turbine with partial span pitch control, in: Société Roumaine de Mathématique appliquées et Industrielles - Oraéda, Roumanie, 1994.

[10] S.G. Voutsinas, V.A. Riziotis, Dynamic Stall and 3D Effects, Technical Report, National Technical University of Athens - JOU2-CT93-0345, 1996.

[11] V.A. Riziotis, S.G. Voutsinas, Dynamic stall modelling on airfoils based on strong viscous–inviscid interaction coupling, Int. J. Numer. Meth. Fluids 56 (2) (2008) 185–208, doi:10.1002/fld.1525.

[12] T. Sebastian, M. Lackner, Development of a free vortex wake method code for offshore floating wind turbines, Renew. Energy 46 (0) (2012) 269–275, doi:10.1016/j.renene.2012.03.033. URL http://www.sciencedirect.com/science/article/pii/S0960148112002315.

[13] M.S. Maza, S. Preidikman, F.G. Flores, Unsteady and non-linear aeroelastic analysis of large horizontal-axis wind turbines, Int. J. Hydrog. Energy 39 (16) (2014) 8813–8820, doi:10.1016/j.ijhydene.2013.12.028. URL http://www.sciencedirect.com/science/article/pii/S0360319913029613.

[14] A. Leonard, Vortex methods for flow simulation, J. Comput. Phys. 37 (3) (1980) 289–335, doi:10.1016/0021-9991(80)90040-6. URL http://www.sciencedirect.com/science/article/B6WHY-4DD1RFV-1T/2/68d5c3a0ce2f03f137353c200b9df3e8, arXiv: http://www.annualreviews.org/doi/pdf/10.1146/annurev.fl.17.010185.002211.

[15] D.G. Crighton, The kutta condition in unsteady flow, Annu. Rev. Fluid Mech. 17 (1) (1985) 411–445, doi:10.1146/annurev.fl.17.010185.002211.

[16] D. O'Doherty, A. Mason-Jones, C. Morris, T. O'Doherty, C. Byrne, P. Prickett, R. Grosvenor, Interactions of marine turbines in close proximity, in: Proceedings of the 9th European Wave and Tidal Energy Conference (EWTEC), Southampton, UK, 2011.

[17] P. Ploumhans, G. Winckelmans, Vortex methods for high resolution simulations of viscous flow past bluff-bodies of general geometry, J. Comput. Phys. 165 (2000) 354–406.

[18] P. Ploumhans, G. Winckelmans, J. Salmon, A. Leonard, M. Warren, Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Application to the sphere at Re=300, 500 and 1000, J. Comput. Phys. 178 (2002) 427–463.

[19] P. Poncet, Analysis of an immersed boundary method for three-dimensional flows in vorticity formulation, J. Comput. Phys. 228 (19) (2009) 7268–7288, doi:10.1016/j.jcp.2009.06.023. URL http://www.sciencedirect.com/science/article/pii/S0021999109003465.

[20] M. Malandain, N. Maheu, V. Moureau, Optimization of the deflated conjugate gradient algorithm for the solving of elliptic equations on massively parallel machines, J. Comput. Phys. 238 (0) (2013) 32–47, doi:10.1016/j.jcp.2012.11.046. URL http://www.sciencedirect.com/science/article/pii/S0021999112007280.

[21] J.R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Lecture Note, Carnegie Mellon University, 1994. URL https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf, (accessed 01.09.16).

[22] 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. Statist. Comput. 13 (2) (1992) 631–644, doi:10.1137/0913035.

[23] P. Degond, S. Mas-Gallic, The weighted particle method for convection-diffusion equations. Part I: The case of an isotropic viscosity, Math. Comp. 53 (188) (1989) 485–507, doi:10.2307/2008716.

[24] J. Choquin, S. Huberson, Particles simulation of viscous flow, Comput. Fluids 17 (2) (1989) 397–410, doi:10.1016/0045-7930(89)90049-2. URL http://www.sciencedirect.com/science/article/pii/0045793089900492.

[25] J.D. Eldredge, A. Leonard, T. Colonius, A general deterministic treatment of derivatives in particle methods, J. Comput. Phys. 180 (2) (2002) 686–709, doi:10.1006/jcph.2002.7112. URL http://www.sciencedirect.com/science/article/B6WHY-46G47HY-F/2/a01a8d80f16f9d8f1275850e03158789.

[26] E. Rivoalen, S. Huberson, The particle strength exchange method applied to axisymmetric viscous flows, J. Comput. Phys. 168 (2) (2001) 519–526, doi:10.1006/jcph.2001.6712. URL http://www.sciencedirect.com/science/article/pii/S0021999101967129.

[27] F.-J. Mustieles, L'équation de Boltzmann des semiconducteurs. Étude mathématique et simulation numérique, École Polytechnique, 1990 Ph.D. thesis.

[28] P. Degond, F. Mustieles, A deterministic approximation of diffusion equations using particles, SIAM J. Sci. Stat. Comput. 11 (1990) 293–310.

[29] Y. Ogami, T. Akamatsu, Viscous flow simulation using the discrete vortex model–the diffusion velocity method, Comput. Fluids 19 (3–4) (1991) 433–441, doi:10.1016/0045-7930(91)90068-S. URL http://www.sciencedirect.com/science/article/pii/004579309190068S.

[30] S. Kempka, J. Strickland, A Method to Simulate Viscous Diffusion of Vorticity by Convective Transport of Vortices at A non-solenoidal Velocity, Technical Report SAND–93-1763, Sandia Laboratory, 1993. URL http://www.osti.gov/energycitations/servlets/purl/10190654-dLAMwy/native/.

[31] P. Mycek, G. Pinon, G. Germain, Élie Rivoalen, A self-regularising DVM–PSE method for the modelling of diffusion in particle methods, Comptes Rendus Mécanique 341 (9–10) (2013) 709–714, doi:10.1016/j.crme.2013.08.002. URL http://www.sciencedirect.com/science/article/pii/S1631072113001137.

[32] P. Mycek, G. Pinon, G. Germain, E. Rivoalen, Formulation and analysis of a diffusion-velocity particle model for transport-dispersion equations, Comput. Appl. Math. 35 (2) (2016) 447–473, doi:10.1007/s40314-014-0200-5.

[33] F. Hauville, Optimisation des méthodes de calculs d'écoulements tourbillonnaires instationnaires, Université du Havre, 1996 Ph.D. thesis. URL http://tel.archives-ouvertes.fr/tel-00125000/PDF/These1996_FH.pdf.

[34] L.M. Milne-Thomson, Theoretical hydrodynamics, Dover Books on Physics Series, Fifth ed., Dover, 1968. URL http://books.google.fr/books?id=cXcfyei9H4MC.

[35] J.L. Hess, Calculation of Potential Flow about Arbitrary Three-Dimensional Lifting Bodies, Defense Technical Information Center, 1969. URL http://books.google.fr/books?id=NKPFOwAACAAJ.

[36] J.L. Hess, Calculation of Potential Flow About Arbitrary Three-Dimensional Lifting Bodies, Technical Report MDC J5679-01, Douglas Aircraft Company, McDonnell Douglas Corporation, 1972.

[37] B. Cantaloube, C. Rehbach, Calcul des intégrales de la méthode des singularités, La Recherche aérospatiale 1 (1986) 15–22. URL http://cat.inist.fr/?aModele=afficheN&cpsidt=7901335.

[38] G.S. Winckelmans, A. Leonard, Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows, J. Comput. Phys. 109 (2) (1993) 247–273, doi:10.1006/jcph.1993.1216. URL http://www.sciencedirect.com/science/article/B6WHY-45P11X4-9/2/18db4eee303171d83ce200e0170c4543.

[39] J.T. Beale, A. Majda, High order accurate vortex methods with explicit velocity kernels, J. Comput. Phys. 58 (2) (1985) 188–208, doi:10.1016/0021-9991(85)90176-7. URL http://www.sciencedirect.com/science/article/B6WHY-4DDR2KW-47/2/a1fa01ee4805464c4e45e0a3dededa59.
[40] G. Coulmy, Formulation des effets de singularités–Seconde partie : Singularités en domaine tridimensionnel, Notes et documents LIMSI 85-6, LIMSI, 1988.
[41] J. Bousquet, Aérodynamique : Méthode des singularités, Collection La chevêche, Cépaduès Editions, 1990. URL http://books.google.fr/books?id=J8X3PAAACAAJ.
[42] P. Mycek, B. Gaurier, G. Germain, G. Pinon, E. Rivoalen, Experimental study of the turbulence intensity effects on marine current turbines behaviour. Part I: One single turbine, Renew. Energy 66 (0) (2014) 729–746, doi:10.1016/j.renene.2013.12.036. URL http://www.sciencedirect.com/science/article/pii/S096014811400007X.
[43] R. Malki, I. Masters, A. Williams, T. Croft, The influence on tidal stream turbine spacing on performance, in: Proceedings of 9th European Wave and Tidal Energy Conference (EWTEC), Southampton, UK, 2011.
[44] I. Masters, A.J. Williams, T.N. Croft, R. Malki, On the performance of axially aligned tidal stream turbines using a blade element disk approach, in: Proceedings of 2nd Oxford Tidal Energy Workshop, 2013.
[45] V. Rokhlin, Rapid solution of integral equations of classical potential theory, J. Comput. Phys. 60 (2) (1985) 187–207.
[46] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (2) (1987) 325–348.
[47] N. Nishimura, Fast multipole accelerated boundary integral equation methods, Appl. Mech. Rev. 55 (4) (2002) 299–324.
[48] Y. Liu, Fast Multipole Boundary Element Method: Theory and Applications in Engineering, Cambridge University press, 2009.
[49] Y. Kervella, G. Germain, B. Gaurier, J.-V. Facq, T. Bacchetti, Mise en évidence de l'importance de la turbulence aambiant sur les effets d'interaction entre hydroliennes, in: XIIIèmes Journées Nationales Génie Côtier –Génie Civil, 2014.