Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2017-0196

Vol. **24**, No. 5, pp. 1503-1522
November 2018

Scalable Domain Decomposition Algorithms for Simulation of Flows Passing Full Size Wind Turbine

Rongliang Chen[1], Zhengzheng Yan[1], Yubo Zhao[1] and Xiao-Chuan Cai[2,*]

[1] *Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, P.R. China.*
[2] *Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA.*

Received 14 September 2017; Accepted (in revised version) 30 November 2017

Abstract. Accurate numerical simulation of fluid flows around wind turbine plays an important role in understanding the performance and also the design of the wind turbine. The computation is challenging because of the large size of the blades, the large computational mesh, the moving geometry, and the high Reynolds number. In this paper, we develop a highly parallel numerical algorithm for the simulation of fluid flows passing three-dimensional full size wind turbine including the rotor, nacelle, and tower with realistic geometry and Reynolds number. The flow in the moving domain is modeled by unsteady incompressible Navier-Stokes equations in the arbitrary Lagrangian-Eulerian form and a non-overlapping sliding-interface method is used to handle the relative motion of the rotor and the tower. A stabilized moving mesh finite element method is introduced to discretize the problem in space, and a fully implicit scheme is used to discretize the temporal variable. A parallel Newton-Krylov method with a new domain decomposition type preconditioner, which combines a non-overlapping method across the rotating interface and an overlapping Schwarz method in the remaining subdomains, is applied to solve the fully coupled nonlinear algebraic system at each time step. To understand the efficiency of the algorithm, we test the algorithm on a supercomputer for the simulation of a realistic 5MW wind turbine. The numerical results show that the newly developed algorithm is scalable with over 8000 processor cores for problems with tens of millions of unknowns.

AMS subject classifications: 65F10, 65F08, 68W10, 76D05, 76U05

Key words: Wind turbine aerodynamics, 3D unsteady incompressible flows, domain decomposition method, fully implicit methods, parallel computing, unstructured finite element method.

*Corresponding author. *Email addresses:* rl.chen@siat.ac.cn (R. Chen), zz.yan@siat.ac.cn (Z. Yan), yb.zhao@siat.ac.cn (Y. Zhao), cai@cs.colorado.edu (X.-C. Cai)

1 Introduction

Wind power is becoming more popular as a renewable energy. The global wind energy council's data shows that the worldwide installed wind power capacity has grown exponentially during the last decade and will continue to grow at a high rate [16]. Wind turbines are the main facility designed to exploit the wind energy. As the demand increases, the industry is moving in the direction of large-scale designs, such as the Vestas V164-8.0MW offshore turbine [40], whose blade length is 80 meters. In the design process, high resolution aerodynamic simulation plays an important role and the computing is challenging because of the large computational domain, the high Reynolds number, and the relative motion of the computational subdomains. Computation at such scales requires large-scale parallel computers and scalable parallel algorithms.

In the last decades, most of the wind turbine aerodynamic simulation research focused on low fidelity methods, such as the blade element momentum method [19,20,33], based on the blade element theory and the momentum theory, which is simple to implement and computationally inexpensive, but is unable to adequately resolve the details of the complex flow structures. Recently, with the rapid development of supercomputers, some high fidelity simulation methods based on the 3D unsteady Navier-Stokes equations are proposed. In 2002, Sorensen et al. [38] introduced a framework based on the Reynolds-Averaged Navier-Stokes model, discretized with an implicit finite volume method for the 3D wind turbine rotor aerodynamic study, where the tower is ignored in the calculation. Bazilevs et al. [4,5,21] combined the large eddy simulation, finite element method, and fully implicit time integration to study the fluid-structure interaction issues of the wind turbine rotor. In simulating wind turbine aerodynamics, only a handful of researchers considered a full wind turbine system, in which the rotor, nacelle, and tower are all included. This is due to the additional computational challenges associated with the simulation of objects in relative motion. For the full wind turbine system, Bazilevs et al. proposed an ALE-VMS technique coupled with a non-overlapping sliding-interface method and a non-uniform rational B-splines based method in [22,23]; Li et al. [32] investigated the detached eddy method with semi-implicit temporal discretization and finite difference spatial discretization based on the dynamic overset grids for the grid deformation and relative motions. Some more studies for full wind turbine systems can be found in [17,45]. Most of the works just mentioned focused on the model accuracy of the methods, not on the parallel scalability which is very important in order to obtain high resolution simulations.

In high fidelity simulations, in order to obtain sufficiently accurate solutions, fine computational meshes are needed, thus requiring large scale parallel computers for their memory capacity and processing speed. It is clear by now that the increase of computing power is no longer from faster processor cores, but from the increase of the number of processor cores. In this study, we focus on developing a scalable parallel method for the simulation of 3D unsteady incompressible flows around a full wind turbine system involving rotor and tower. To deal with objects with relative motion, generally speak-

ing, there are two approaches; i.e., the shear-slip mesh update method which keeps the mesh connectivity by introducing a regular, typically remeshing layer between the two domains in relative motion [7], and the non-overlapping sliding-interface method where an interpolation is used to transfer information between the two sets of variables defined on the interface [6]. In this paper, we choose the latter approach. For the discretization, a stabilized unstructured finite element method is used in the spatial domain and a fully implicit finite difference scheme is employed in the temporal direction. A parallel Newton-Krylov-Schwarz (NKS) method [8, 25, 26] is proposed to solve the large sparse nonlinear system at each time step, where an inexact Newton method is employed as the nonlinear solver, a Krylov subspace method is used as the linear Jacobian system solver in the Newton steps, and a new domain decomposition method, that combines a non-overlapping method across the rotating interface and an overlapping Schwarz method for the other subdomains, is used as a preconditioner to accelerate the convergence of the linear solver. NKS has been used to solve a wide range of problems, such as PDE constrained optimization problems [10,43], fluid-structure interaction problems [2,29,31,42], elasticity problems [30], shallow water equations [44] and so on. But it has not been studied for problems considered in this paper which involve moving domain and relative motions in a single configuration. We mainly investigate the parallel performance of the solution methods, including their robustness, stability and scalability.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the mathematical model, and the discretization of the mathematical model is discussed in Section 3. In Section 4, the Newton-Krylov-Schwarz algorithm is introduced, and some numerical results are presented in Section 5. Some concluding remarks are given in Section 6.

2 Mathematical model

As shown in Fig. 1, the computational domain consists of a stationary domain Ω_s and a rotating domain Ω_r. The rotor is in Ω_r and the domain Ω_s encloses the tower. The flow comes in from the right side Γ_{inlet}, goes out at the left side Γ_{outlet}. We follow the standard approach to introduce an rotating interface $\Gamma_{interface}$ to separate the stationary and rotating domains, use separate fluid flow models in the two subdomains, and connect them using a boundary condition on the interface. For simplicity, we assume the entire domain Ω_r rotates in the x-z plane and does not move in the y direction (the direction of the tower is parallel to the z-axis and the rotor faces to the y-axis).

For the stationary domain, the standard incompressible Navier-Stokes equations read as:

$$\begin{aligned} \rho\left(\frac{\partial \mathbf{u}_s}{\partial t}+\mathbf{u}_s\cdot\nabla\mathbf{u}_s\right)+\nabla\cdot\sigma_s=\mathbf{f} \quad & \text{in } \Omega_s\times(0,T), \\ \nabla\cdot\mathbf{u}_s=0 \quad & \text{in } \Omega_s\times(0,T), \\ \mathbf{u}_s=\mathbf{0} \quad & \text{on } \Gamma_{wall}\times(0,T), \end{aligned}$$

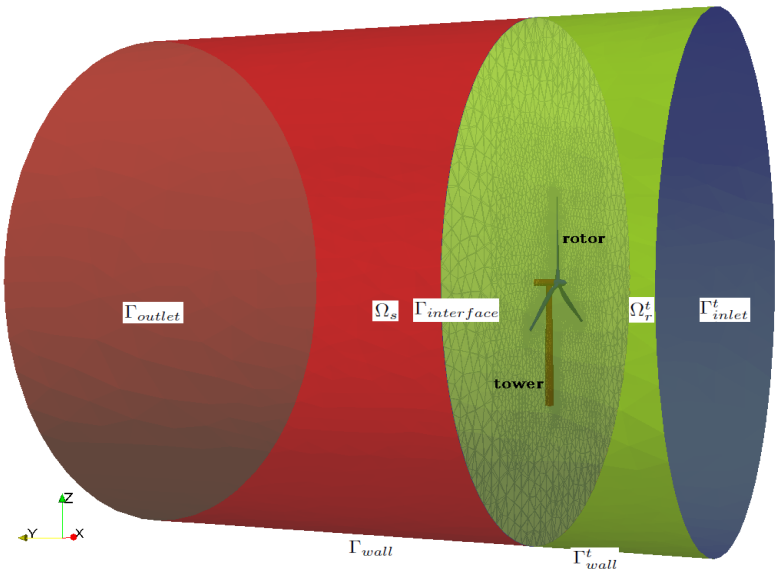

Figure 1: The computational domain consists of two subdomains. One (Ω_r^t) moves at the same angular velocity with the rotor and the other one (Ω_s) stays still with the tower.

$$\begin{aligned} \sigma_s \cdot \mathbf{n} &= 0 \quad \text{on } \Gamma_{outlet} \times (0, T), \\ \mathbf{u}_s &= \mathbf{u_0} \quad \text{in } \Omega_s \text{ at } t = 0, \end{aligned} \tag{2.1}$$

where $\mathbf{n}$ is the unit outward normal, $\sigma_s = -p_s \mathbf{I} + \mu\left(\nabla \mathbf{u}_s + (\nabla \mathbf{u}_s)^T\right)$ is the Cauchy stress tensor, and $\mathbf{u}_s$, p_s are the velocity and pressure of the flow. For the rotating domain, the governing equations are the incompressible Navier-Stokes equations in the arbitrary Lagrangian-Eulerian (ALE) form [12,24]. Let $\mathbf{Y}$ be the ALE coordinate, then the equations read as:

$$\begin{aligned} \rho\left(\left.\frac{\partial \mathbf{u}_r}{\partial t}\right|_{\mathbf{Y}} + (\mathbf{u}_r - \omega)\cdot\nabla \mathbf{u}_r\right) + \nabla \cdot \sigma_r &= \mathbf{f} \quad \text{in } \Omega_r^t \times (0, T), \\ \nabla \cdot \mathbf{u}_r &= 0 \quad \text{in } \Omega_r^t \times (0, T), \\ \mathbf{u}_r &= \mathbf{g} \quad \text{on } \Gamma_{inlet}^t \times (0, T), \\ \mathbf{u}_r &= \mathbf{0} \quad \text{on } \Gamma_{wall}^t \times (0, T), \\ \mathbf{u}_r &= \mathbf{u_0} \quad \text{in } \Omega_r^t \text{ at } t = 0, \end{aligned} \tag{2.2}$$

where $\omega = \frac{\partial \mathbf{x}}{\partial t}$ is the velocity of the moving domain, $\sigma_r = -p_r \mathbf{I} + \mu\left(\nabla \mathbf{u}_r + (\nabla \mathbf{u}_r)^T\right)$, $\mathbf{u}_r$ and p_r are the velocity and pressure, and $\left.\frac{\partial \mathbf{u}_r}{\partial t}\right|_{\mathbf{Y}}$ indicates that the time derivative is to be taken with respect to the ALE coordinate $\mathbf{Y}$. $\mathbf{u_0}$ is a given initial condition which is zero in our test cases.

In addition to (2.1) and (2.2), two coupling conditions are needed on $\Gamma_{interface}$ to insure the continuity of the flow across the interface [6]. The first condition is the continuity of

the velocity,

$$\mathbf{u}_s = \mathbf{u}_r \qquad \text{on} \quad \Gamma_{interface}, \tag{2.3}$$

and the second condition is the continuity of the traction force,

$$\sigma_s \cdot \mathbf{n}_s = -\sigma_r \cdot \mathbf{n}_r \qquad \text{on} \quad \Gamma_{interface}. \tag{2.4}$$

Here $\mathbf{n}_s$ and $\mathbf{n}_r$ are the unit outward normals to the stationary and rotating subdomains, respectively.

3 Fully-implicit finite element discretization

We consider a weak formulation of the coupled incompressible Navier-Stokes systems (2.1) and (2.2) [6]. Let $\mathcal{U}_\cdot(\Omega_\cdot)$ and $\mathcal{U}_\cdot^0(\Omega_\cdot)$ denote the trial and weighting function spaces of the velocity, and $\mathcal{P}_\cdot$ for the pressure, where "$\cdot$" represents s or r which refers to as the stationary subdomain or the rotating subdomain, respectively. Then, the weak form of the Navier-Stokes equations takes the form: Find $\mathbf{u}_s \in \mathcal{U}_s$, $\mathbf{u}_r \in \mathcal{U}_r$, $p_s \in \mathcal{P}_s$, and $p_r \in \mathcal{P}_r$, such that

$$\mathbf{B}_s(\mathbf{u}_s, p_s; \Phi_s, \psi_s) + \mathbf{B}_r(\mathbf{u}_r, p_r; \Phi_r, \psi_r) - \mathbf{F}_s(\Phi_s, \psi_s) - \mathbf{F}_r(\Phi_r, \psi_r) = \mathbf{0} \tag{3.1}$$

holds for any $\Phi_s \in \mathcal{U}_s^0$, $\Phi_r \in \mathcal{U}_r^0$, $\psi_r \in \mathcal{P}_r$, and $\psi_s \in \mathcal{P}_s$, where

$$\begin{aligned}
\mathbf{B}_s(\mathbf{u}_s, p_s; \Phi_s, \psi_s) =& \rho \int_{\Omega_s} \frac{\partial \mathbf{u}_s}{\partial t} \cdot \Phi_s d\Omega_s + \mu \int_{\Omega_s} \nabla \mathbf{u}_s : \nabla \Phi_s d\Omega_s + \rho \int_{\Omega_s} (\mathbf{u}_s \cdot \nabla) \mathbf{u}_s \cdot \Phi_s d\Omega_s \\
& - \int_{\Omega_s} p_s^h \nabla \cdot \Phi_s d\Omega_s + \int_{\Omega_s} (\nabla \cdot \mathbf{u}_s) \varphi_s d\Omega_s + \int_{\Gamma_{interface}} \Phi_s \cdot (\sigma_s \cdot \mathbf{n}_s) d\Gamma, \\
\mathbf{B}_r(\mathbf{u}_r, p_r; \Phi_r, \psi_r) =& \rho \int_{\Omega_r^t} \left. \frac{\partial \mathbf{u}_r}{\partial t} \right|_{\mathbf{Y}} \cdot \Phi_r d\Omega_r^t + \mu \int_{\Omega_r^t} \nabla \mathbf{u}_r : \nabla \Phi_r d\Omega_r^t + \rho \int_{\Omega_r^t} (\mathbf{u}_r - \omega) \cdot \nabla) \mathbf{u}_r \cdot \Phi_r d\Omega_r^t \\
& - \int_{\Omega_r^t} p \nabla \cdot \Phi_r d\Omega_r^t + \int_{\Omega_r^t} (\nabla \cdot \mathbf{u}_r) \varphi_r d\Omega_r^t + \int_{\Gamma_{interface}} \Phi_r \cdot (\sigma_r \cdot \mathbf{n}_r) d\Gamma, \\
\mathbf{F}_r(\Phi_r, \psi_r) =& \int_{\Omega_r^t} \mathbf{f} \cdot \Phi_r d\Omega_r^t, \quad \mathbf{F}_s(\Phi_s, \psi_s) = \int_{\Omega_s} \mathbf{f} \cdot \Phi_s d\Omega_s.
\end{aligned}$$

Note that the coupling condition (2.4) is implicitly enforced as part of (3.1) by the relation [42]

$$\int_{\Gamma_{interface}} \Phi_s \cdot (\sigma_s \cdot \mathbf{n}_s) d\Gamma + \int_{\Gamma_{interface}} \Phi_r \cdot (\sigma_r \cdot \mathbf{n}_r) d\Gamma = 0.$$

We use a $P_1 - P_1$ finite element method to discretize Eq. (3.1) in the spatial domain. Since the $P_1 - P_1$ element method does not satisfy the Ladyzenskaja-Babuska-Brezzi (LBB)

condition, additional stabilization terms are needed in the formulation with equal-order interpolation of the velocity and the pressure as described in [3, 15, 41, 42].

In the spatial discretization, the interface condition (2.3) needs to be handled carefully since the mesh from the left side (denoted as $\Gamma^{hs}_{interface}$) and the right side (denoted as $\Gamma^{hr}_{interface}$) do not match on $\Gamma_{interface}$; see Fig. 2, we assume the rotating side as the master side whose value is exact, and other side is the slave side whose value is obtained by interpolation. We use a radial basis function (RBF) interpolation method [11, 14] to enforce the interface condition, that is

$$\mathbf{u}_s(x_s^k) = \sum_{i=1}^{m} w_i \phi\left(\| x_s^k - x_r^i \| \right) \mathbf{u}_r(x_r^i),$$

where x_s^k and x_r^i are the mesh points on the tower and the rotor side respectively, and $m = 4$ is the number of points used for the interpolation, that is all the nodes of the element containing the slave node. w_i $(i = 1, \cdots, m)$ are the interpolation weights for each point that are obtained by solving a m dimensional linear system analytically. The function $\phi(r) = \sqrt{1+(\epsilon r)^2}$ is the multiquadric RBF basis function and ϵ is a parameter which is proportional to $1/h$ where h is the mesh size [14]. In this paper, h is chosen as the maximum distance between two adjacent points in $\{x_r^i, i = 1, \cdots, m\}$.

Remark 3.1. Standard interpolation using the finite element basis functions does not work well for several reasons:

1. Since we use a $P_1 - P_1$ finite element method, we can only have a first-order interpolation if we use the finite element basis function (which is a linear function) as the interpolation basis function, while the RBF based interpolation can have a higher order. For cases with large Reynolds number, the flow is quite complex, the first-order interpolation is not accurate enough to insure the continuity across the interface.

2. Some nodes on $\Gamma^{hs}_{interface}$ are outside of $\Gamma^{hr}_{interface}$; see, for example, node B on the boundary of $\Gamma^{hs}_{interface}$ in Fig. 2 is outside of $\Gamma^{hr}_{interface}$. Standard interpolation would miss part of the solution since the node is not in any elements, while RBF is able to capture this since it is based on the distance to a node, not restricted by the elements.

Remark 3.2. The interpolation is carried out implicitly. In other words, the values on both sides of the interface are unknowns at the time of the interpolation. The actual values are obtained by solving the coupled system at each time step.

Remark 3.3. In our current implementation, the values from Ω_r are taken as the master side and values on Ω_s side are obtained by the interpolation. Switching the master side does not change the accuracy, but it does change the structure of the Jacobian matrix.

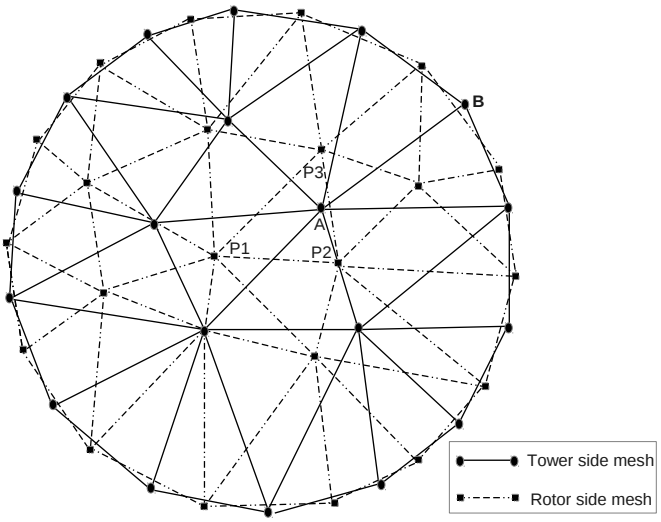

Figure 2: An example of the meshes on the interface $\Gamma_{interface}$. The mesh with solid lines is denoted as $\Gamma^{hs}_{interface}$ and the mesh with dashed lines is denoted as $\Gamma^{hr}_{interface}$. The value of $\mathbf{u}_s$ at point A is interpolated by the values of $\mathbf{u}_r$ at points P1, P2, and P3.

Remark 3.4. Since the slave nodes on the tower side move from element to element of the rotor side, and on each time step we need to identify the element that each slave node belongs to. To make the search more efficient, we scatter the rotor side interface boundary information $\Gamma^{hr}_{interface}$ to all processor cores, that is every processor core has all the information of $\Gamma^{hr}_{interface}$. We then carry out the search in parallel based on the tower side interface boundary.

For the temporal discretization, we use a fully implicit second-order backward differentiation formula (BDF2) with a fixed time step size Δt. For a given semi-discretized system

$$\frac{d\mathbf{X}}{dt} = \mathbf{L}(\mathbf{X}),$$

the formula is defined as

$$\frac{\mathbf{X}^n - \frac{4}{3}\mathbf{X}^{n-1} + \frac{1}{3}\mathbf{X}^{n-2}}{\Delta t} = \frac{2}{3}\mathbf{L}(\mathbf{X}^n). \tag{3.2}$$

In BDF2, at each time step (the n^{th} step), we need to solve a large sparse nonlinear algebraic system, denoted as

$$\mathbf{F}^n(\mathbf{X}^n) = \mathbf{0}, \tag{3.3}$$

using solutions from the previous time steps $\mathbf{X}^{n-1}$ and $\mathbf{X}^{n-2}$ (the solution of the $(n-1)^{th}$ and $(n-2)^{th}$ time step), to obtain the solution of the n^{th} time step $\mathbf{X}^n$, which includes the

nodal values of the velocity and pressure for both flow domains. For the first time step, a backward Euler scheme is used since there is only one previous solution available.

4 Monolithic Newton-Krylov-Schwarz algorithm

The nonlinear system (3.3) is solved by a Newton-Krylov-Schwarz method which uses an inexact Newton method [13] as the nonlinear solver, a Krylov subspace method GMRES [36] as the linear solver at each Newton step, and a non-standard Schwarz method as the preconditioner to accelerate the convergence of the linear solver. The main components of the algorithm read as follows:

- *Find an initial condition* $\mathbf{U}^0$ *for both domains and set* $n=0$ *(n refers to as the* n^{th} *time step)*
- *For* $n=1, 2, \cdots$, *do*
  - *Rotate the domain* $(\Omega_r^{n-1} \to \Omega_r^n)$ *and its mesh* $\mathcal{T}_h^n$:
    The coordinate of each mesh point at the current time step
    $\mathbf{x}^n$ *is obtained by rotating the initial mesh point* $\mathbf{x}^0$

$$\mathbf{x}^n = \begin{bmatrix} \cos(\omega n\Delta t) & 0 & -\sin(\omega n\Delta t) \\ 0 & 1 & 0 \\ \sin(\omega n\Delta t) & 0 & \cos(\omega n\Delta t) \end{bmatrix} \mathbf{x}^0$$

  - *Setup the initial guess for Newton* $\mathbf{U}_0^n = \mathbf{U}^{n-1}$ *and let* $k=0$
  - *For* $k=1, 2, \cdots$, *until convergence, do*
    - *Compute the nonlinear function* $\mathbf{F}^n(\mathbf{U}_{k-1}^n)$ *and form the corresponding Jacobian matrix* $\mathbf{J}_k^n = \nabla \mathbf{F}^n(\mathbf{U}_{k-1}^n)$ *analytically including all terms*
    - *Find* $\mathbf{d}_k^n$ *such that*

$$\| \mathbf{J}_k^n (\mathbf{M}_k^n)^{-1} (\mathbf{M}_k^n \mathbf{d}_k^n) + \mathbf{F}^n(\mathbf{U}_{k-1}^n) \| \leq \eta \| \mathbf{F}^n(\mathbf{U}_{k-1}^n) \| \tag{4.1}$$

    - *Set* $\mathbf{U}_k^n = \mathbf{U}_{k-1}^n + \tau_k^n \mathbf{d}_k^n$
  - *Set* $\mathbf{U}^n = \mathbf{U}_k^n$

Here $(\mathbf{M}_k^n)^{-1}$ is a preconditioner to be introduced below, η is the relative tolerance for the linear solver, $\mathbf{U} := (\mathbf{u}_r, p_r, \mathbf{u}_s, p_s)$, and $0 \leq \tau_k^n \leq 1$ is the step length obtained by a linesearch method. Note that the interface condition is applied in the function evaluation and the Jacobian calculation.

The Jacobian matrix $\mathbf{J}_k^n$ includes three main blocks, two for the two flow subdomains, and one for the interpolation on the interface. If the unknowns are ordered field by field

and all the unknowns related to the rotating interface are put together at the middle, then the structure of the Jacobian matrix $\mathbf{J}_k^n$ takes the form

$$\begin{pmatrix} \mathbf{K_r} & -\mathbf{Q_r} & \mathbf{D}_{r1} & 0 & 0 & 0 \\ \mathbf{Q_r^T} & \mathbf{R_r} & \mathbf{D}_{r2} & 0 & 0 & 0 \\ \mathbf{D}_{r1}^T & \mathbf{D}_{r2}^T & \mathbf{D}_{rI} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{D}_{rsI} & \mathbf{D}_{sI} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{D}_{s1} & \mathbf{K_s} & -\mathbf{Q_s} \\ 0 & 0 & 0 & \mathbf{D}_{s2} & \mathbf{Q_s^T} & \mathbf{R_s} \end{pmatrix},$$

where $\mathbf{K_s}$, $\mathbf{Q_s}$, $\mathbf{R_s}$, and $\mathbf{Q_s^T}$ are for the velocity and pressure, respectively, in the stationary flow domain, $\mathbf{K_r}$, $\mathbf{Q_r}$, $\mathbf{R_r}$, and $\mathbf{Q_r^T}$ are for the rotating flow domain, and $\mathbf{D}_{rsI}$, $\mathbf{D}_{s1}$, $\mathbf{D}_{s2}$, $\mathbf{D}_{r1}$, and $\mathbf{D}_{r2}$ correspond to the interface. $\mathbf{R_s}$ and $\mathbf{R_r}$ are from the stabilization terms. This ordering is easy to understand, but if used in the code, the performance is not good. In order to improve the cache performance we use a point-block ordering which orders the variables mesh point by mesh point and all the variables associated with the same mesh point are ordered together. An example of the matrix structure is show in Fig. 3. Note that the non-zero pattern of the matrix is non-symmetric. This is because we apply the interpolation from only one side.

The most time-consuming step of the algorithm is the solution of the large, sparse, and nonsymmetric linear system (4.1) which is solved by a Krylov subspace method (GMRES) in this paper. To accelerate the convergence of GMRES, we introduce a non-standard restricted additive Schwarz preconditioner [9], which begins with a partition of the finite element mesh in the entire computational domain Ω (including Ω_s and Ω_r^0) into N nonoverlapping subdomains Ω_l $(l=1,\cdots,N)$; see Fig. 4, and then extending each subdomain Ω_l to an overlapping subdomain Ω_l^δ by including δ layers of elements belonging to its neighbors. Note that, the nonoverlapping partitions of the two fluid domains are obtained by a single call of the partitioning software, as a result, the load is balanced, but the partitions of the two fluid domains are independent of each other because their associated graphs are not connected. When we extend the nonoverlapping subdomains to overlapping subdomains, we do not allow the partition to go through the interface, which means that the neighboring subdomains associated with the interface are nonoverlapping. It is, of course, possible to obtain a standard overlapping partition which will go through the interface, and define the standard overlapping domain decomposition preconditioners based on this partition. But then at each time step, we would have to do a global search of the interface elements to find the neighboring subdomains and add the overlaps because the neighboring subdomains change due to the rotation, which will take extra time for computation and communication. In our new domain decomposition method, since we ignore the overlapping parts on the other side of the interface, the partition for Ω_r^0 is usable for all Ω_r^t (for any $t>0$), because the mesh topology does not change when the rotating part of the domain changes from Ω_r^0 to Ω_r^t. We let the total number of subdomains N equal to the number of processor cores n_p. In each overlapping subdomain, we define a local Jacobian matrix $(\mathbf{J}_k^n)_l$ which is obtained by taking the derivatives

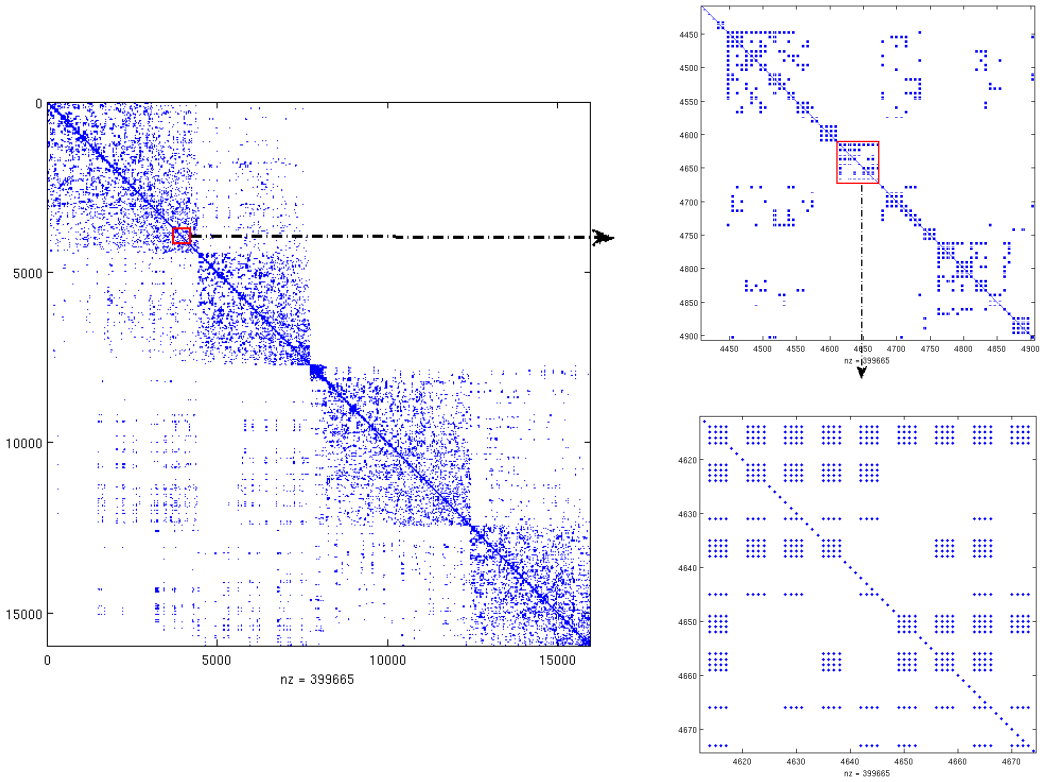

Figure 3: An example of the non-zero structure of the Jacobian matrix plotted by the Matlab function "spy". The non-zero elements are shown in blue. The 4×4 dense blocks are due to the point-block ordering.

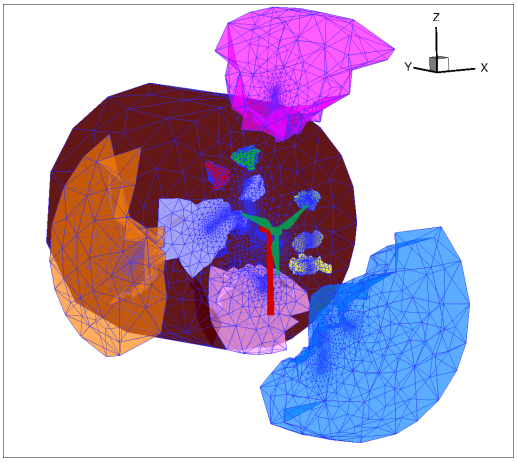

Figure 4: A sample partition obtained by ParMETIS. Since we use nonuniform meshes in our test cases, the mesh near the wind turbine is relative finer than the mesh in the far field. All subdomains have nearly the same number of elements for the purpose of load balancing.

of the discretized equations (3.1) in subdomain Ω_l^δ with homogeneous Dirichlet boundary conditions on the interior boundary $\partial\Omega_l^\delta \backslash \partial\Omega$, and the physical boundary conditions on $\partial\Omega_l^\delta \bigcap \partial\Omega$. The non-standard restricted additive Schwarz preconditioner is defined as the summation of the local preconditioners $\mathbf{B}_l^{-1}$ of $(\mathbf{J}_k^n)_l$ as

$$(\mathbf{M}_k^n)^{-1} = \sum_{l=1}^{n_p} (R_l^0)^T \mathbf{B}_l^{-1} R_l^\delta, \tag{4.2}$$

where the restriction operators R_l^δ and R_l^0 are matrices which map the global vector of unknowns to those belonging to Ω_l^δ and Ω_l respectively, by simply extracting the unknowns that lie inside the subdomain [9]. Since $\mathbf{B}_l^{-1}$ is used as a preconditioner here, it can be solved exactly or approximately, for example, by LU or incomplete LU factorization (ILU) methods. LU factorization is a commonly used method but is computationally expensive and requires a lot of memory when the local matrix $(\mathbf{J}_k^n)_l$ is large. To improve the efficiency of such factorization, we use an ILU method [35] which reduces the computational cost and memory requirement by dropping some nonzero elements in some predetermined nondiagonal positions. In this paper, we use a point-block version of ILU as the subdomain solver, where we group all physical components associated with a mesh point as a block and always use an exact inverse for this small block which is done before the ILU is carried out, in addition, all components in the block are either kept or dropped together.

Remark 4.1. When constructing the preconditioner, we ignore the interface condition in (2.3) and (2.4). The reason is that the variables $\mathbf{u}_s$ and $\mathbf{u}_r$ are on different processor cores and it saves communication time if we simply set $\mathbf{u}_s = \mathbf{0}$ instead of $\mathbf{u}_s = \mathbf{u}_r$.

Remark 4.2. Since the Jacobian matrix $\mathbf{J}_k^n$ is used for computing the Newton search direction and preconditioner, it can be derived exactly or approximately. There are several approaches to calculate $\mathbf{J}_k^n$, such as the automatic differentiation [34], the finite difference method, and the analytic differentiation of all terms of the nonlinear function or some of the terms (for example, by dropping the stabilization terms). After lots of testing, we find that the analytic differentiation of all terms is most efficient in terms of the number of iterations and the total compute time.

5 Numerical experiments

In this section, we report some numerical experiments using the proposed algorithm. Our solver is implemented on top of the Portable Extensible Toolkit for Scientific computation (PETSc) [1]. The unstructured tetrahedral mesh is generated with ANSYS. The mesh partition for parallel computing is obtained with ParMETIS [28]. The results showed in this section are obtained on the Tianhe-2 supercomputer at the National Supercomputer Center in Guangzhou, China. The compute node consists of a dual six-core Intel

Xeon X5650@2.76GHz processor and has 24GB of memory. The stopping conditions for the nonlinear and linear solvers are that when the residuals of the nonlinear and linear equations are reduced by a factor of 10^{-6} and 10^{-4}, respectively. In GMRES, the restart $r=30$ [35]. For the point-block ILU, the block size is 4 by 4 (each mesh point has 4 unknowns: three velocity components and one pressure).

5.1 The rotor only calculation

Our first test case is to compute the flow passing a wind turbine rotor without the tower, see Fig. 5 for the detailed geometry and the computational domain. In this calculation, only the Navier-Stokes equations in the moving domain (2.2) are solved.

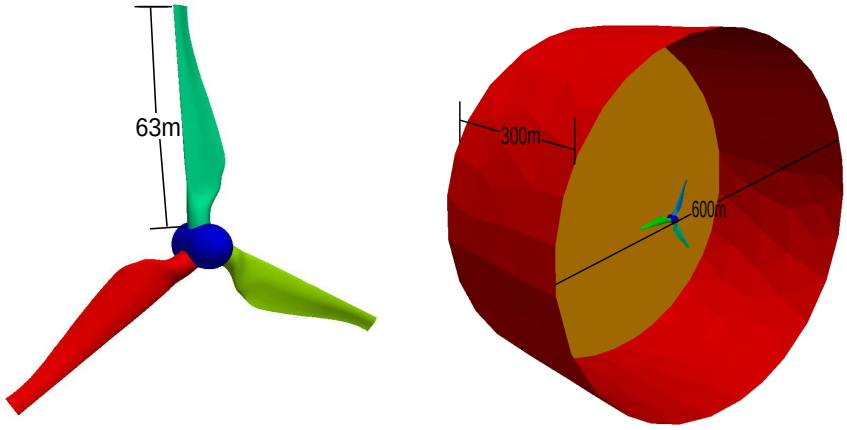

Figure 5: A three-blade wind turbine with NREL S807 root region and NREL S806 tip region [18] (left) and the computational domain (right).

In this numerical experiment, we set the far field wind speed to be uniform at $15m/s$ and the rotor speed at $22rpm$ (revolutions per minute). For the fluid, we set the kinematic viscosity $\mu=1.831\times10^{-5}kg/(ms)$ and the density $\rho=1.185kg/m^3$. Fig. 6 shows the computed flow field at four different times obtained on a mesh with about 1.1×10^7 elements with a fixed time step size $\Delta t=0.01s$.

The left column of Fig. 6 shows the velocity magnitude of the flow at four different time points. From this figure, we see that the largest velocity magnitudes are located in the vicinity of the rotor tips and a low speed region is formed behind the rotor, indicating the wake region, because of the incoming airflow is blocked by the rotor. The maximum width of the wake region is about half of the rotor diameter. The vortical structure of the flow is shown in the right column of Fig. 6.

5.2 The full turbine system calculation

Our second test case is the simulation of a full turbine system, see Fig. 7, including the rotor (blades and hub), nacelle, and tower. The land-based tower is assumed to be rigid,

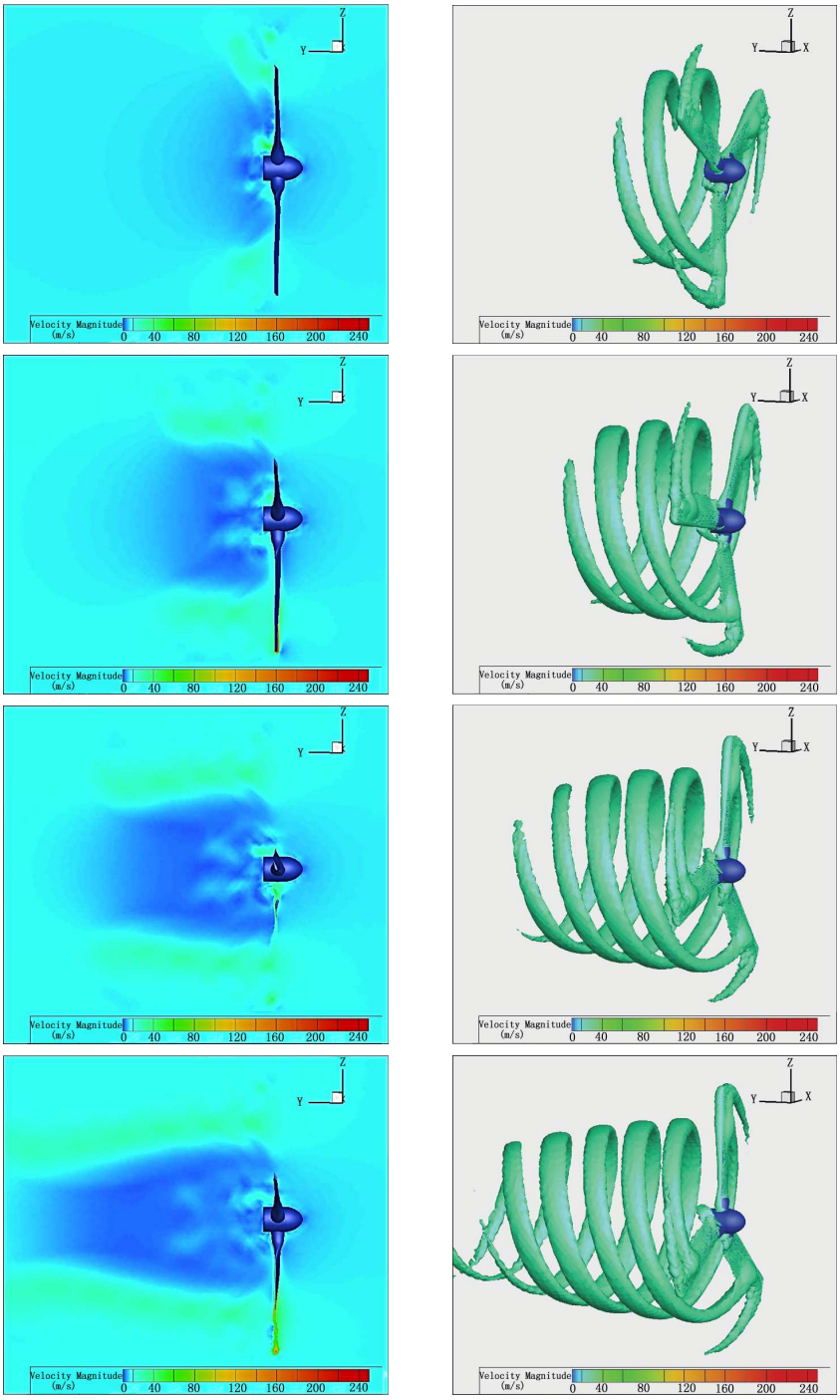

Figure 6: The distribution of the velocity magnitude (left) and the isosurface (right) of the velocity field at four different moments: $t=2.0$, $t=6.0$, $t=8.0$, $t=10.0$ (from top to bottom).

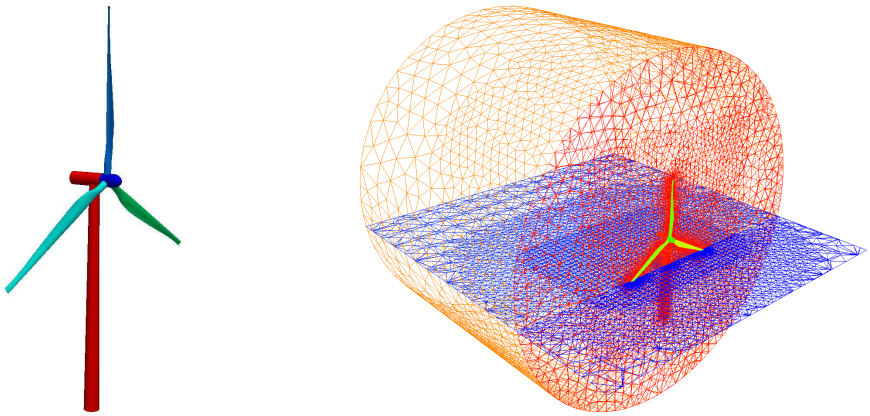

Figure 7: The geometry model (left) and a sample computational mesh (right) for the full wind turbine calculation.

and has a base diameter of $6m$ and a top diameter of $3.87m$. The tower height is $87.6m$ and the hub height is $90m$ [27]. The rotor used in this test case is the same as the one in the first test case.

We set the wind speed to be $11.4m/s$, the rotor speed to be $12.1rpm$, and the air properties to be the same as in the first test case [23]. Non-uniform meshes are used for this calculation where a finer mesh is used at the vicinity of the wind turbine in order to capture the details of the complex flow structure, for example, for the 1.2×10^7 case, the mesh size at the vicinity of the wind turbine is about $0.07m$ and it is about $20m$ in the farfield. The computed velocity distribution at t=10.0s from two different view of points is shown in Fig. 8. The 3D streamline and 2D streamline on three different planes $z=0$, $z=-10$, and $z=-20$ at $t=10.0$ are plotted in Fig. 9, which shows that a complex 3D vortex is generated behind the wind turbine and several vortices appear in the downwind area of the blade and tower.

5.3 Parallel performance and robustness

The parallel performance of our algorithm is shown in Table 1 where (as in all the tables) "DOF" refers to the degree of freedoms, "n_p" refers to as the number of processor cores which is the same as the number of subdomains, "Newton" is the average number of Newton iterations per time step, "GMRES" is the number of GMRES iterations per Newton step for solving the Jacobian system, "Time" is the average compute time (in seconds) per time step, "Mem" is the maximum memory usage of each processor core (in megabits), and "Speedup" and "Ideal" are the actual speedup and ideal speedup, respectively. The time step size for this section is $\Delta t=0.01$. This table shows that the number of GMRES iterations increases slightly with the increase of the number of subdomains, which is often the case in one-level Schwarz methods [39]. The reason is that the overlap between adjacent subdomains is not sufficient for global information transfer when the

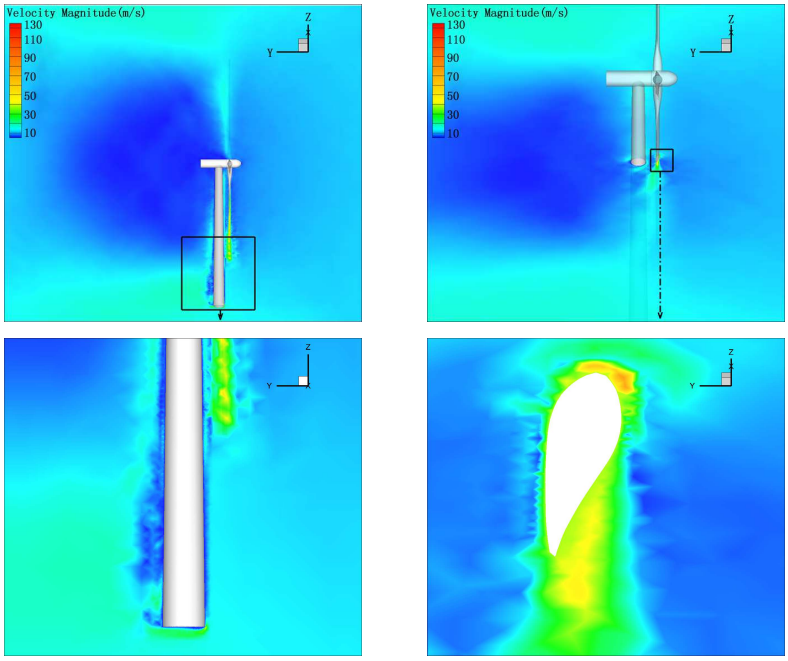

Figure 8: The velocity magnitude contour plot of the flow from different points of view. The left figure is the plane $x=0$ and the right figure is the plane $z=-20$ (the direction of the tower is parallel to the z-axis).

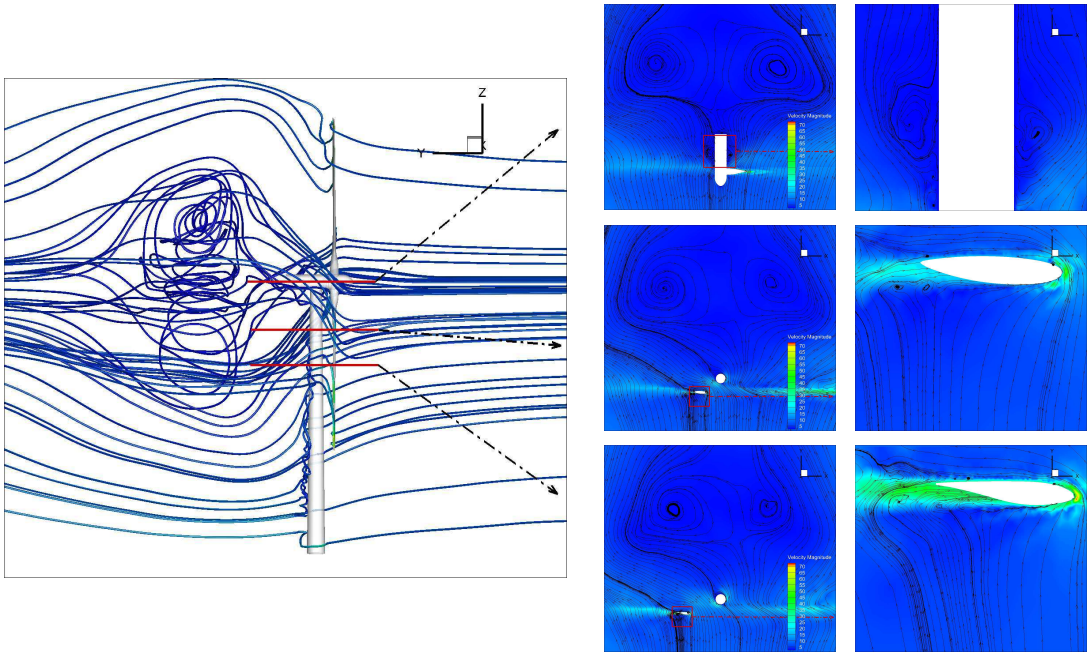

Figure 9: The 3D streamline distribution around the wind turbine (left). The 2D velocity magnitude contour and streamline on three different planes $z=0$, $z=-10$, and $z=-20$ (right) and some local zoom-in figures (far right).

Table 1: Parallel scalability and efficiency for the full wind turbine system simulation.

n_p	Newton	GMRES	Time (s)	Mem (Mb)	Speedup	Ideal	Efficiency
DOF = 7.5×10^6							
512	2.3	38.0	19.0	988	1	1	100%
1024	2.3	38.2	10.8	490	1.8	2	90%
2048	2.3	40.3	6.8	273	2.8	4	70%
4096	2.3	41.5	4.5	147	4.2	8	53%
DOF = 1.2×10^7							
1024	2.3	54.7	20.1	786	1	1	100%
2048	2.3	57.2	11.7	425	1.7	2	85%
4096	2.3	58.5	7.4	227	2.7	4	68%
8192	2.3	63.2	5.3	96	3.8	8	48%

number of subdomain increases, which makes the conditioning of the Jacobian system worse. From this table, we also see that the parallel efficiency "Efficiency" of the algorithm is nearly 70% when the number of processor cores is up to 4096 and the efficiency decreases with the increase of the number of processor cores, because the communication time among processor cores increasingly dominates the total compute time.

In the overlapping domain decomposition method, there are several parameters to consider, such as the overlapping parameter and the ILU level in the subdomain problem solver, and all of them may have some impact on the overall performance. Table 2 shows the effect of the overlapping parameter for the standard DD method and the non-standard DD method introduced in this paper, where larger overlapping means faster convergence in terms of the number of iterations, but the total compute time grows because the size of the subdomain problems increases. Here the standard DD method refers to the method that the overlap is added based on the matrix instead of the mesh. In the standard DD method, some of the overlapping subdomains cross over the interface between the stationary and rotating domains. Table 2 shows that the non-standard DD method is faster than the standard one, especially when the overlapping size is large, and the memory requirement of the standard DD method is larger than the non-standard one. Note that for the case $\delta = 0$, the RAS preconditioner equals to the Jacobi preconditioner [37]. As mentioned in Section 4, a point-block ILU method is used as the subdomain solver. In the ILU method, the fill-in levels l, is used to balance the strength of the preconditioner and the computational cost. Larger l means fewer fill-in elements are dropped during the factorization, a stronger preconditioner is thus obtained, which implies faster convergence. But the additional arithmetic operations may increase the total compute time. The effect of various choices of the ILU fill-in levels and a comparison of the point-wise and point-block ILU method are shown in Table 3. It is clear that the point-block version is much better than the point-wise version in terms of the compute time, due to the much improved cache performance. Table 4 shows the robustness of

Table 2: The effect of various choices of the overlapping parameter δ for the full wind turbine system simulation. Here $\delta=0.5$ means the neighboring subdomains only share a layer of faces. $DOF=1.2\times10^7$ and $n_p=1024$.

	Standard DD				Non-standard DD			
δ	Newton	GMRES	Time (s)	Mem (Mb)	Newton	GMRES	Time (s)	Mem (Mb)
0	—	>300	—	—	—	>300	—	—
0.5	—	—	—	—	2.3	82.0	22.5	625
1	2.3	52.8	22.5	857	2.3	59.0	22.4	709
2	2.3	44.8	25.8	1112	2.3	54.7	23.0	786
3	2.3	41.5	44.5	1485	2.3	52.0	23.9	843
4	2.3	40.1	68.3	1946	2.3	48.0	24.5	942

Table 3: Tests for various choices of the ILU fill-in levels l for the full wind turbine system simulation with fixed problem size ($DOF=1.2\times10^7$) and fixed overlapping size $\delta=2$ and $n_p=1024$.

	Point-wise ILU				Point-block ILU			
l	Newton	GMRES	Time (s)	Mem (Mb)	Newton	GMRES	Time (s)	Mem (Mb)
0	—	>300	—	—	—	>300	—	—
1	2.3	67.8	548.6	43	2.3	67.6	17.6	377
2	2.3	55.7	576.9	86	2.3	54.7	21.6	786
3	2.3	48.0	568.6	151	2.3	48.0	34.7	1393

Table 4: The robustness of the algorithm with respect to the Reynolds number Re for the full wind turbine system simulation. The problem size is $DOF=1.2\times10^7$, the overlapping size $\delta=2$, and $n_p=1024$.

Re	Newton	GMRES	Time (s)
4.0×10^5	3.3	71.7	35.6
8.0×10^5	3.3	72.2	35.7
1.5×10^6	3.4	72.9	35.6
3.0×10^6	3.4	74.6	35.9
6.0×10^6	3.4	74.8	36.3
1.2×10^7	3.4	78.4	37.6
2.3×10^7	3.5	83.6	40.7
4.5×10^7	4.0	94.8	49.7
9.0×10^7	4.6	93.5	56.4

the proposed algorithm with respect to the Reynolds number. The number of Newton and GMRES iterations and the compute time increase reasonably with the increase of the Reynolds number.

6 Concluding remarks

In this paper, a Newton-Krylov-Schwarz based parallel algorithm, combined with a finite element method on unstructured moving meshes for spatial discretization and a fully implicit method for temporal discretization, was developed for the aerodynamic simulation of a full wind turbine system including rotor, nacelle, and tower. The computational domain is divided into two subdomains, one contains the rotating rotor and one for the far field. The meshes across the subdomains do not match. To insure the continuity of the flow across the rotating interface, a RBF based second-order interpolation was developed to pass function values between the two subdomains in relative motion. To solve the large sparse Jacobian system, we introduced a non-standard domain decomposition method which uses overlapping subdomains away from the rotating interface and non-overlapping subdomains near the rotating interface, and numerical experiments confirm the effectiveness of the preconditioning technique. The numerical results show that the proposed algorithm converges well in terms of the nonlinear and linear number of iterations for the problem with realistic geometry and Reynolds number. Good parallel efficiency is obtained for a problem with over 1.2×10^7 unknowns and 8192 processor cores. The overall approach is robust and scalable and has the potential to be used for solving larger problems for higher fidelity simulations on supercomputers.

Acknowledgments

The research was supported in part by the Special Project on High-performance Computing under the National Key R&D Program (No. 2016YFB0200601), the NSFC under 11401564, 61531166003, 11571100 and U1501501, and the Shenzhen basic research grant under JCYJ20170307165328836 and JCYJ20160331193229720.

References

[1] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, H. Zhang, PETSc Users Manual, http://www.mcs.anl.gov/petsc, Argonne National Laboratory, 2017.

[2] A. Barker, X.-C. Cai, Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling, J. Comput. Phys. 229 (2010) 642-659.

[3] Y. Bazilevs, V.M. Calo, T.J.R. Hughes, Y. Zhang, Isogeometric fluid-structure interaction: theory, algorithms, and computations, Comput. Mech. 43 (2008) 3-37.

[4] Y. Bazilevs, M.C. Hsu, I. Akkerman, S. Wright, K. Takizawa, B. Henicke, T. Spielman, T.E. Tezduyar, 3D simulation of wind turbine rotors at full scale. Part I: geometry modeling and aerodynamics, Int. J. Numer. Methods Fluids 65 (2011) 207-235.

[5] Y. Bazilevs, M.C. Hsu, J. Kiendl, R. Wuchner, K.U. Bletzinger, 3D simulation of wind turbine rotors at full scale. Part II: fluid-structure interaction modeling with composite blades, Int. J. Numer. Methods Fluids 65 (2011) 236-253.

[6] Y. Bazilevs, T.J.R. Hughes, NURBS-based isogeometric analysis for the computation of flows about rotating components, Comput. Mech. 43 (2008) 143-150.
[7] M. Behr, T.E. Tezduyar, The shear-slip mesh update method, Comput. Methods Appl. Mech. Eng. 174 (1999) 261-274.
[8] X.-C. Cai, W.D. Gropp, D.E. Keyes, R.G. Melvin, D.P. Young, Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation, SIAM J. Sci. Comput. 19 (1998) 246-265.
[9] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 21 (1999) 792-797.
[10] R. Chen, X.-C. Cai, Parallel one-shot Lagrange-Newton-Krylov-Schwarz algorithms for shape optimization of steady incompressible flows, SIAM J. Sci. Comput. 34 (2012) B584-B605.
[11] S. Deparis, D. Forti, A. Quarteroni, A rescaled localized radial basis function interpolation on non-Cartesian and nonconforming grids, SIAM J. Sci. Comput. 36 (2014) A2745-A2762.
[12] F. Duarte, R. Gormaz, S. Natesan, Arbitrary Lagrangian-Eulerian method for Navier-Stokes equations with moving boundaries, Comput. Methods Appl. Mech. Eng. 193 (2004) 4819-4836.
[13] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1996) 16-32.
[14] B. Fornberg, J. Zuev, The Runge phenomenon and spatially variable shape parameters in RBF interpolation, Comput. & Math. Appl. 54 (2007) 379-398.
[15] L.P Franca, S.L. Frey, Stabilized finite element method: II. The incompressible Navier-Stokes equations, Comput. Methods Appl. Mech. Eng. 99 (1992) 209-233.
[16] Global Wind Energy Outlook 2012, http://www.gwec.net/publications/global-wind-energy-outlook/.
[17] S. Gomez-Iradi, R. Steijl, G.N. Barakos, Development and validation of a CFD technique for the aerodynamic analysis of HAWT, J. Solar Energy Engng. 131 (2009) 1-13.
[18] GrabCAD webpage: https://grabcad.com/.
[19] M. Hansen, Aerodynamics of Wind Turbines, 2nd Edition, Erthscan, London, 2008.
[20] A.C. Hansen, C.P. Butterfield, Aerodynamics of horizontal axis wind turbines, Ann. Rev. Fluid Mech. 25 (1993) 115-149.
[21] M.C. Hsu, I. Akkerman, Y. Bazilevs, High-performance computing of wind turbine aerodynamics using isogeometric analysis, Comput. & Fluids 49 (2011) 93-100.
[22] M.C. Hsu, I. Akkerman, Y. Bazilevs, Finite element simulation of wind turbine aerodynamics: validation study using NREL Phase VI experiment, Wind Energy 17 (2013) 461-481.
[23] M.C. Hsu, Y. Bazilevs, Fluid-structure interaction modeling of wind turbines: simulating the full machine, Comput. Mech. 50 (2012) 821-833.
[24] T.J.R. Hughes, W.K. Liu, T.K. Zimmermann, Lagrangian-Eulerian finite element formulation for incompressible viscous flows, Comput. Methods Appl. Mech. Eng. 29 (1981) 329-349.
[25] F.-N. Hwang, X.-C. Cai, A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations, J. Comput. Phys. 204 (2005) 666-691.
[26] F.-N. Hwang, C.-Y. Wu, X.-C. Cai, Numerical simulation of three-dimensional blood flows using domain decomposition method on parallel computer, J. Chinese Soc. Mech. Eng. 31 (2010) 199-208.
[27] J. Jonkman, S. Butterfield, W. Musial, G. Scott, Definition of a 5-MW reference wind turbine for offshore system development, Technical Report NREL/TP-500-38060, National Renew-

able Energy Laboratory, Golden, CO, 2009.
[28] G. Karypis, METIS/ParMETIS webpage, University of Minnesota, 2015, http://glaros.dtc.umn.edu/gkhome/views/metis.
[29] F. Kong and X.-C. Cai. Scalability study of an implicit solver for coupled fluid-structure interaction problems on unstructured meshes in 3D, Int. J. High Perform. Comput. Appl. (2016) 1094342016646437.
[30] F. Kong and X.-C. Cai. A highly scalable multilevel Schwarz method with boundary geometry preserving coarse spaces for 3D elasticity problems on domains with complex geometry, SIAM J. Sci. Comput., 38 (2016), C73-C95.
[31] F. Kong and X.-C. Cai. A scalable nonlinear fluidstructure interaction solver based on a Schwarz preconditioner with isogeometric unstructured coarse spaces in 3D, J. Comput. Phys., 340 (2017), 498-518.
[32] Y. Li, K.J. Paik, T. Xing, P.M. Carrica, Dynamic overset CFD simulations of wind turbine aerodynamics, Renew. Energy 37 (2012) 285-298.
[33] H. Madsen, R. Mikkelsen, S. Oye, C. Bak, J. Johansen, A detailed investigation of the blade element momentum (BEM) model based on analytical and numerical results and proposal for modifications of the BEM model, J. Phys.: Conf. Ser. 75 (2007) 012016.
[34] R. Neidinger, Introduction to automatic differentiation and MATLAB object-oriented programming, SIAM Rev. 52 (2010) 545-563.
[35] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, Boston, 1996.
[36] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Comput. Stat. Comput. 7 (1986) 856-869.
[37] B. Smith, P. Bjørstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, New York, 1996.
[38] N.N. Sorensen, J.A. Michelsen, S. Schreck, Navier-Stokes predictions of the NREL phase VI rotor in the NASA Ames 80 ft $\times$ 120 ft wind tunnel, Wind Energy 5 (2002) 151-169.
[39] A. Toselli, O. Widlund, Domain Decomposition Methods: Algorithms and Theory, Springer-Verlag, Berlin, 2005.
[40] Vestas V164-8.0 MW offshore wind turbine, http://www.vestas.com/en/wind-power-plants/procurement/turbine-overview/v164-8.0-mw-offshore.aspx.
[41] C.H. Whiting, K.E. Jansen, A stabilized finite element method for the incompressible Navier-Stokes equations using a hierarchical basis, Int. J. Numer. Methods Fluids 35 (2001) 93-116.
[42] Y. Wu, X.-C. Cai, A fully implicit domain decomposition based ALE framework for three-dimensional fluid-structure interaction with application in blood flow computation, J. Comput. Phys. 258 (2014) 524-537.
[43] H. Yang, E. Prudencio, X.-C. Cai, Fully implicit Lagrange-Newton-Krylov-Schwarz algorithms for boundary control of unsteady incompressible flows, Int. J. Numer. Methods Engng. 91 (2012) 644-665.
[44] C. Yang, J. Cao, X.-C. Cai, A fully implicit domain decomposition algorithm for shallow water equations on the cubed-sphere, SIAM J. Sci. Comput. 32 (2010) 418-438.
[45] F. Zahle, N.N. Sorensen, J. Johansen, Wind turbine rotor-tower interaction using an incompressible overset grid method, Wind Energy 12 (2009) 594-619.