Computer Physics Communications 270 (2022) 108170

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Large eddy simulation of the wind flow in a realistic full-scale urban community with a scalable parallel algorithm ☆

Zhengzheng Yan [a,b], Rongliang Chen [a,b,*], Xiao-Chuan Cai [c]

[a] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
[b] Shenzhen Key Laboratory for Exascale Engineering and Scientific Computing, Shenzhen 518055, China
[c] Department of Mathematics, University of Macau, Macau, China

ARTICLE INFO

Article history:
Received 16 July 2020
Received in revised form 11 August 2021
Accepted 13 September 2021
Available online 15 September 2021

Keywords:
Urban wind flow
Parallel computing
Domain decomposition method
Large eddy simulation
Unstructured mesh

ABSTRACT

Turbulence and large computational domains make urban wind flow simulation a computationally challenging problem. To reduce the total simulation time and efficiently use today's new generation of supercomputers with massive processors, scalable parallel solvers are needed. In this work, we present a scalable domain decomposition method based 3D incompressible Navier-Stokes solver for the simulation of unsteady, complex wind flows around urban communities. A large eddy simulation with Smagorinsky modeling is employed for the simulation of the turbulent wind flows. On a fine unstructured grid, we discretize the spatial and temporal dimensions using a stabilized $P_1 - P_1$ finite element method and an implicit backward Euler finite difference method, respectively. The resulting large-scale nonlinear algebraic system at each timestep is solved by a Newton-Krylov-Schwarz algorithm in parallel. The solver is first validated by a benchmark case where the numerical results are compared with measured data from a wind tunnel. Then, the wind flow field of a realistic actual-scale urban community with a group of buildings in the downtown area of Shenzhen, China, is studied. The numerical results show that the flow field matches with the experimental data for the benchmark problem, and some reasonable, detailed, and complex flow structures are obtained for the urban community simulation case. The scalability of the solver is studied on the TianHe-2A and Sunway TaihuLight supercomputers, and the solver scales up to 16,384 processor cores for a grid with over 20 million elements, which implies that the solver has the potential to perform fast and high-fidelity simulations of large-scale wind flows in complicated computational domains.

1. Introduction

Urban wind flow analysis is related to many architectural and environmental applications. Conventionally, the main motivation for researching the wind flow around urban areas is to understand the urban wind dynamics to estimate wind loads on structures [1]. In addition to wind vulnerability, wind flow also impacts air pollutant concentrations and floating object movements, and knowledge of the flow field is the basis of understanding and modeling the dispersion of pollutants [2,3]. Moreover, in recent years, wind flow simulation has been widely used in supporting the sustainable design considerations of cities and buildings, such as natural ventilation design and pedestrian comfort [4,5].

Numerical methods with computational fluid dynamics (CFD) have become vital tools in urban wind flow analysis [6,7]. Numerical investigations on urban flow are challenging due to the large, complex computational domain and the nature of turbulent wind. Thus, geometric simplification and various approximate turbulence models are usually applied in wind flow simulation studies. With respect to the computational geometry, the spatially inhomogeneous geometry is ignored, and the urban architectures are generally considered as simplified rough surfaces, porous media, or standard shapes [8–11]. However, the geometrical details of the buildings have been proven to significantly affect the flow motion around them, which pushed some researchers to use realistic real-scale building models [12]. With respect to the turbulence model, researchers have developed several approaches, such as Reynolds-averaged Navier-Stokes (RANS) [13], unsteady Reynolds-averaged Navier-Stokes (URANS) [14,15], large eddy simulation (LES) [16–18], and hybrid URANS/LES (also termed the detached eddy simulation (DES)) [19,20], to capture the features of the wind flow based on different methods for modeling small eddies. The

☆ The review of this paper was arranged by Prof. David W. Walker.

* Corresponding author at: Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China.
E-mail addresses: zz.yan@siat.ac.cn (Z. Yan), rl.chen@siat.ac.cn (R. Chen), xccai@um.edu.mo (X.-C. Cai).

https://doi.org/10.1016/j.cpc.2021.108170
0010-4655/

LES model generally exhibits superior performance compared with RANS and URANS because a large part of the unsteady turbulent flow is actually resolved [21].

Wind flow simulation is computationally expensive, especially for LES. Correspondingly, powerful hardware resources and parallel scalable solvers are essential because of the enormous computational requirements. During the last few years, supercomputers with massive processors have significantly improved the computational capabilities. Parallel scalable solvers are useful because they not only enable the solution of large-scale urban wind flow problems but also lead to dramatic compression of the solution time. However, most of the urban flow simulations use commercial codes [22] which can only scale up to hundreds of cores [23,24].

There are some large-scale urban flow simulations, for example, based on the finite difference method and equidistant horizontal Cartesian grid, PALM (the parallelized large-eddy simulation model) [25] scales up to 20,000 processor cores, and it has been widely used for large scale atmospheric and urban flow simulations [26–28]. Onodera et al. [29] simulated the flow in a 10 km $\times$ 10 km area of Tokyo utilizing 4032 GPU cards on a $10,080 \times 10,240 \times 512$ mesh based on the lattice Boltzmann method (LBM). Ahmad et al. [30] use 900 GPU cards to simulate the intensity of wind in a 19.2 km $\times$ 4.8 km $\times$ 1.0 km built-up areas at the pedestrian level. Lenz et al. [31] simulated the wind flow in a 500 m $\times$ 500 m $\times$ 152 m urban canopy of Basel on a GPU card with 3584 CUDA cores. All of these simulations are based on the structured grid which has advantages such as less memory usage, easy to implement, and better parallel performance, but the disadvantage is that it can not be used to do simulations with complicated geometries [32].

The aim of the present study is to introduce a scalable parallel solver for solving the incompressible Navier-Stokes equations on unstructured grids, and apply it to perform urban wind flow simulations with a LES turbulence model. Several strong and weak scaling tests are studied on the top two supercomputers in China, which shows that the proposed solver scales up to thousands of processor cores with an acceptable parallel efficiency. The layout of the paper is as follows. First, in Section 2, numerical methods are introduced in terms of governing equations and parallel algorithms. Second, Section 3 describes two numerical experiments, including a benchmark problem and the wind flow over a realistic full-scale urban community, and the parallel performance of this solver is presented and discussed, corresponding to the objectives of the present study. Finally, some conclusions and remarks are provided in Section 4.

2. Mathematical model and solution algorithm

2.1. Governing equations

In this paper, we employ a LES approach with the Smagorinsky model [33] to simulate the 3D transient flow. The Navier-Stokes equations for incompressible flow with the LES model can be described as

$$\begin{aligned} \frac{\partial \bar{\mathbf{u}}}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}} &= -\frac{1}{\rho}\nabla \bar{p} + \nabla \cdot (2\nu \bar{\mathbf{S}}) - \nabla \cdot \boldsymbol{\tau} \quad \text{in} \quad \Omega, \\ \nabla \cdot \bar{\mathbf{u}} &= 0 \qquad\qquad\qquad\qquad\quad \text{in} \quad \Omega, \end{aligned} \tag{1}$$

where $\bar{\mathbf{u}}$ is the filtered velocity, ρ is the fluid density, ν is the viscosity, $\bar{p}$ denotes the filtered pressure, and $\Omega \subset R^3$ is the bounded computational domain. The bar lines indicate the grid-scale filtering operator, and $\bar{\mathbf{S}}$ is the filtered strain rate tensor defined as

$$\bar{\mathbf{S}} = \frac{1}{2}(\nabla \bar{\mathbf{u}} + (\nabla \bar{\mathbf{u}})^T).$$

$\tau = \overline{\mathbf{u}\mathbf{u}} - \bar{\mathbf{u}}\bar{\mathbf{u}}$ is the subgrid scale (SGS) stress tensor, which accounts for the effect of unresolved small-scale eddies on the dynamics at large scales. To close the system, we need to introduce a method to model the SGS stress τ. In this paper, τ is modeled by introducing an eddy-viscosity concept such that

$$\tau^d \equiv \tau - \left(\frac{1}{3}\mathrm{tr}(\boldsymbol{\tau})\right)\mathbf{I} = -2\nu_t \bar{\mathbf{S}}. \tag{2}$$

Here, τ^d is the deviatoric part of the SGS stress, and ν_t is the Smagorinsky eddy viscosity, which is defined as

$$\nu_t = (C_s \bar{\Delta})^2 |\bar{\mathbf{S}}|,$$

and C_s is the Smagorinsky constant, with values varying from 0.1 to 0.2. $\bar{\Delta}$ is the filter width, which is fundamentally based on the cube root of the cell volume. $|\bar{\mathbf{S}}|$ is the characteristic filtered strain rate, also known as the strain magnitude, which is defined as $|\bar{\mathbf{S}}| = \sqrt{2\bar{\mathbf{S}} : \bar{\mathbf{S}}} = \sqrt{2\bar{S}_{ij}\bar{S}_{ij}}$. By substituting Eq. (2) into Eq. (1) and defining $\bar{P}$ as the modified filtered pressure

$$\bar{P} = \frac{\bar{p}}{\rho} + \frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}),$$

the governing equations can be rewritten as

$$\begin{aligned} \frac{\partial \bar{\mathbf{u}}}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}} &= -\nabla \bar{P} + \nabla \cdot (2\nu \bar{\mathbf{S}}) + \nabla \cdot (2\nu_t \bar{\mathbf{S}}) \quad \text{in} \quad \Omega, \\ \nabla \cdot \bar{\mathbf{u}} &= 0 \qquad\qquad\qquad\qquad\qquad\quad \text{in} \quad \Omega. \end{aligned} \tag{3}$$

For the boundary conditions, the no-slip boundary condition is applied on the walls Γ_{wall} (including the top of the computational domain, the ground, and the surfaces of building). A Dirichlet boundary condition is placed on the inlet Γ_{inlet}, that is,

$$\bar{\mathbf{u}} = \mathbf{g} \quad \text{on} \quad \Gamma_{inlet}.$$

The stress-free boundary condition is set on the outlet Γ_{outlet}, which reads as

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0} \quad \text{on} \quad \Gamma_{outlet},$$

where $\boldsymbol{\sigma} = -\bar{P}\mathbf{I} + 2(\nu + \nu_t)\bar{\mathbf{S}}$ is the Cauchy stress tensor.

For the initial condition, a divergence-free velocity field is specified over the domain at $t = 0$ s

$$\bar{\mathbf{u}}(\mathbf{x}, 0) = \mathbf{u_0}(\mathbf{x}) \quad \text{in} \quad \Omega \quad \text{at} \quad t = 0.$$

2.2. Spatial and temporal discretization

We apply a $P_1 - P_1$ finite element method for the spatial discretization of the governing equations Eq. (3). The trial and weighting function spaces are defined as:

$$\begin{aligned} \mathcal{U} = \{\mathbf{u}(\cdot,t) \mid \mathbf{u}(\cdot,t) \in [H^1(\Omega)]^3, \quad & \mathbf{u}(\cdot,t) = \mathbf{g} \quad \text{on} \quad \Gamma_{inlet}, \\ & \mathbf{u}(\cdot,t) = \mathbf{0} \quad \text{on} \quad \Gamma_{wall}\}, \\ \mathcal{P} = \{p(\cdot,t) \mid p(\cdot,t) \in L^2(\Omega)\}, & \\ \mathcal{U}_0 = \{\mathbf{u}(\cdot,t) \mid \mathbf{u}(\cdot,t) \in [H^1(\Omega)]^3, \quad & \mathbf{u}(\cdot,t) = \mathbf{0} \ \text{ on } \ \Gamma_{inlet} \cup \Gamma_{wall}\}. \end{aligned}$$

Then, the Galerkin weak form of the LES equations (3) takes the form: find $\bar{\mathbf{u}} \in \mathcal{U}$ and $\bar{P} \in \mathcal{P}$ such that:

$$B_G(\bar{\mathbf{u}}, \bar{P}; \boldsymbol{\omega}, q) = 0, \quad \forall (\boldsymbol{\omega}, q) \in \mathcal{U}_0 \times \mathcal{P}, \tag{4}$$

with

$$B_G(\bar{\mathbf{u}}, \bar{P}; \boldsymbol{\omega}, q) = \left(\frac{\partial \bar{\mathbf{u}}}{\partial t}, \boldsymbol{\omega}\right)_\Omega + (\bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}}, \boldsymbol{\omega})_\Omega - \left(\bar{P}, \nabla \cdot \bar{\boldsymbol{\omega}}\right)_\Omega + \left(2(\nu + \nu_t)\bar{\mathbf{S}}, \nabla \boldsymbol{\omega}\right)_\Omega + (\nabla \cdot \bar{\mathbf{u}}, q)_\Omega .$$

Here, $(\mathbf{f}, \boldsymbol{\omega})_\Omega$ represents the standard scalar inner product $\int_\Omega \mathbf{f} \cdot \boldsymbol{\omega} \mathrm{d}\Omega$ in $L^2(\Omega)$.

An unstructured tetrahedral mesh $\mathcal{T}^h = \{K\}$ is used to delineate the complex geometry of the computational domain, and after discretization the finite dimensional trial and weighting spaces can be established as

$$\begin{aligned}
\mathcal{U}^h &= \{\mathbf{u}^h \mid \mathbf{u}^h \in [C^0(\Omega) \cap H^1(\Omega)]^3, \quad \mathbf{u}^h \mid_{K \in \mathcal{T}^h} \in P_1(K)^3, \\
&\qquad \mathbf{u}^h = \mathbf{g} \quad \text{on} \quad \Gamma_{inlet}, \\
&\qquad \mathbf{u}^h = \mathbf{0} \quad \text{on} \quad \Gamma_{wall}\}, \\
\mathcal{P}^h &= \{p^h \mid p^h \in C^0(\Omega) \cap L^2(\Omega), \quad p^h \mid_{K \in \mathcal{T}^h} \in P_1(K)\}, \\
\mathcal{U}_0^h &= \{\mathbf{u}^h \mid \mathbf{u}^h \in [C^0(\Omega) \cap H^1(\Omega)]^3, \quad \mathbf{u}^h \mid_{K \in \mathcal{T}^h} \in P_1(K)^3, \\
&\qquad \mathbf{u}^h = \mathbf{0} \quad \text{on} \quad \Gamma_{inlet} \cup \Gamma_{wall}\},
\end{aligned}$$

where $C^0(\Omega)$ is the set of all continuous functions defined on Ω and $P_1(K)^3$ is the space of piecewise linear functions.

We introduce a stabilization term [34] since the $P_1 - P_1$ finite element method does not satisfy the Ladyzenskaja-Babuska-Brezzi (LBB) condition. Finally, the semi-discrete finite element system, as shown in Eq. (4), can be expressed as follows: find $\bar{\mathbf{u}}^h \in \mathcal{U}^h$ and $\bar{P}^h \in \mathcal{P}^h$, such that:

$$B(\bar{\mathbf{u}}^h, \bar{P}^h; \boldsymbol{\omega}^h, q^h) = 0, \quad \forall \left(\boldsymbol{\omega}^h, q^h\right) \in \mathcal{U}_0^h \times \mathcal{P}^h$$

with

$$\begin{aligned}
&B(\bar{\mathbf{u}}^h, \bar{P}^h; \boldsymbol{\omega}^h, q^h) \\
&= B_G(\bar{\mathbf{u}}^h, \bar{P}^h; \boldsymbol{\omega}^h, q^h) + \sum_{K \in \mathcal{T}^h} \left(\nabla \cdot \bar{\mathbf{u}}^h, \ \tau_c \nabla \cdot \boldsymbol{\omega}^h\right)_K \\
&\quad + \sum_{K \in \mathcal{T}^h} \Bigg(\frac{\partial \bar{\mathbf{u}}^h}{\partial t} + (\bar{\mathbf{u}}^h \cdot \nabla)\bar{\mathbf{u}}^h + \nabla \bar{P}^h \\
&\quad - 2(\nu + \nu_t)\nabla \cdot \bar{\mathbf{S}}, \tau_m(\bar{\mathbf{u}}^h \cdot \nabla \boldsymbol{\omega}^h + \nabla q^h) \Bigg)_K ,
\end{aligned}$$

where τ_c and τ_m are the stabilization parameters defined as

$$\tau_c = \frac{1}{8\tau_m \mathrm{tr}(\mathbf{G})},$$

$$\tau_m = \left(\sqrt{\frac{4}{(\Delta t)^2} + (\bar{\mathbf{u}} \cdot \mathbf{G} \cdot \bar{\mathbf{u}}) + 36\left(\frac{\mu}{\rho}\right)^2 \mathbf{G} : \mathbf{G}} \right)^{-1},$$

where $\mathbf{G} = \{G_{ij}\} = \left\{ \sum_{k=1}^{3} \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j} \right\}$ is the covariant metric tensor, $\frac{\partial \xi}{\partial x}$ denotes the inverse Jacobian of the mapping between the reference and the physical element, and Δt is the timestep size to be introduced in the temporal discretization.

For temporal discretization, we choose the fully implicit backward Euler finite difference formula, as shown in Eq. (5), which offers some advantages; for example, one can use a large timestep size (because of its unconditioned numerical stability) to save the overall simulation time in large-scale numerical computing.

$$\frac{\mathbf{x}^n - \mathbf{x}^{n-1}}{\Delta t} = S(\mathbf{x}^n), \tag{5}$$

where $S(\mathbf{x}^n)$ is the semidiscretized system of the Navier-Stokes equations (3) except the first term $\frac{\partial \bar{\mathbf{u}}}{\partial t}$ after the spatial discretization and $\mathbf{x}^n = (\mathbf{u}^n, p^n)$ represents the velocity and pressure at the n^{th} timestep. In the fully implicit method, a large, sparse, and nonlinear system must be solved at each timestep to obtain the solution at the next timestep, which is simply denoted as:

$$\mathbf{F}^n(\mathbf{x}^n) = \mathbf{0}. \tag{6}$$

2.3. Domain decomposition method-based parallel solver

In this section, a domain decomposition method (DDM)-based Newton-Krylov-Schwarz (NKS) [35] method is applied to solve the large algebraic system (6) in parallel. DDM is conducive to parallel computing due to the characteristics of the divide and conquer. Specific to this study, an inexact Newton method [36] is applied to solve the nonlinear equations, a preconditioned Krylov subspace method (restarted GMRES) [37] is used to obtain the Newton corrections, and the restricted additive Schwarz (RAS) method [38] is introduced as the preconditioner in the preconditioned Krylov subspace method to accelerate the solver.

In the NKS method, the most expansive step is the RAS preconditioner to be introduced shortly. The RAS method is a kind of overlapping DDM, and it begins with partitioning the computational domain Ω into n_p nonoverlapping subdomains Ω_i, $i = 1, 2, \cdots, n_p$, where n_p is usually equal to the number of processors; then, we obtain the overlapping subdomains Ω_i^δ by extending δ layers of mesh elements from the adjacent subdomains. Each Ω_i^δ can define its own local Jacobian matrix $\mathbf{J}_i$ in a similar way as the method to construct the global Jacobian matrix. To define the RAS method, we need to define the global-to-local restriction operator R_i^δ, which is defined to return all of the degrees of freedom (DOF) belonging to Ω_i^δ from the global vector of unknowns, and a similar restriction operator R_i^0, which corresponds to the nonoverlapping subdomain Ω_i [38]; then, the RAS preconditioner is defined as the summation of the local preconditioner $\mathbf{B}_i^{-1}$ of $\mathbf{J}_i$ in the way

$$\mathbf{M}_{RAS} = \sum_{i=1}^{n_p} (R_i^0)^T \mathbf{B}_i^{-1} R_i^\delta .$$

For the subdomain preconditioner $\mathbf{B}_i^{-1}$, to save computation time, we use a point-block incomplete LU (ILU) factorization [39] of $\mathbf{J}_i$ with some levels of fill-ins instead of the LU factorization method, which is too computationally expensive. As a comparison, we show the parallel performance of the LU and ILU methods as the subdomain solver in the numerical experiments section.

The framework of the parallel Newton-Krylov-Schwarz algorithm for solving the nonlinear system (6) can be summarized as:

Step (a): Let $\mathbf{x}_0^n$ be the initial guess, which is usually the previous timestep solution $\mathbf{x}^{n-1}$. At each Newton step, the new approximate solution $\mathbf{x}_{k+1}^n$ is updated by the following equation:

$$\mathbf{x}_{k+1}^n = \mathbf{x}_k^n + \lambda_k^n s_k^n,$$

where λ_k^n is the step length obtained in Step (b) and $\mathbf{s}_k^n$ is the search direction calculated in Step (c).

Step (b): Calculate the step length λ_k^n by a cubic backtracking line search method [40].

Step (c): Construct the Jacobian matrix $\mathbf{J}_k^n$ of $\mathbf{F}^n$ at $\mathbf{x}_k^n$ and solve the following preconditioned Jacobian system:

$$\mathbf{J}_k^n (\mathbf{M}_k^n)^{-1} \mathbf{y}_k^n = -\mathbf{F}^n(\mathbf{x}_k^n), \tag{7}$$

and then solve

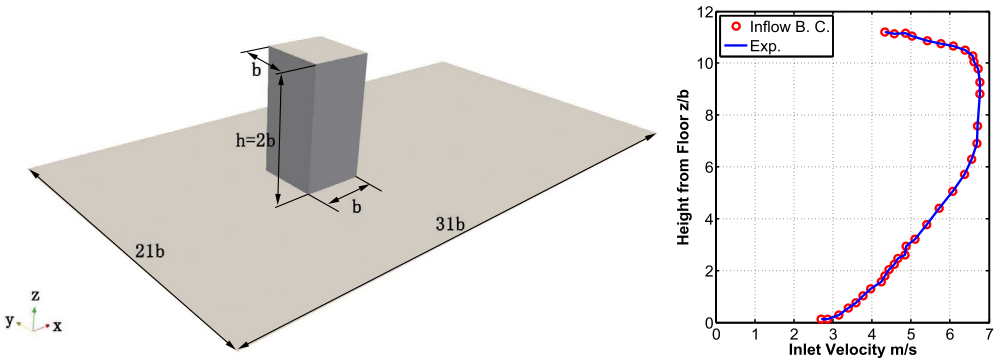


Fig. 1. Model of the benchmark problem (left) and the measured and interpolated inflow boundary condition (right).

$$\mathbf{M}_k^n \mathbf{s}_k^n = \mathbf{y}_k^n,$$

where $(\mathbf{M}_k^n)^{-1}$ *is the RAS preconditioner mentioned above. Since we employ the inexact Newton method in this paper, the linear system (7) is solved inexactly by a restarted GMRES method in the sense of*

$$\|\mathbf{F}^n\left(\mathbf{x}_k^n\right) + \mathbf{J}_k^n (\mathbf{M}_k^n)^{-1} \mathbf{M}_k^n \mathbf{s}_k^n\| \leq \eta_k \|\mathbf{F}^n\left(\mathbf{x}_k^n\right)\|,$$

where η_k *is a parameter to control the accuracy of the solution of the linear system, which is also the stopping condition for the GMRES method.*

Step (d): Check the stopping condition. If it is satisfied, then stop and obtain the solution; otherwise, go to Step (a), update the initial guess $\mathbf{x}_0^n$ *with* $\mathbf{x}_{k+1}^n$ *and let* $k = k + 1$.

In this study, our parallel solver is implemented on the top of the open source package PETSc [41] from Argonne National Lab, and the unstructured meshes are generated with ANSYS ICEM CFD [42] and partitioned with ParMETIS [43] from the University of Minnesota. ParaView [44] is used to visualize the simulated flow fields. For more details of this solver, we refer to our previous paper [45,46] and the references cited therein. All the computations are performed on the top two supercomputers of China, TianHe-2A and Sunway TaihuLight, at Guangzhou and Wuxi, China, respectively. An AMAX high-performance computing platform, which has a dual 8-core Intel E52687@3.1 GHz processor and 128 GB memory, is used for preprocessing and postprocessing.

3. Numerical experiments

In this section, we present two numerical studies, a benchmark problem and a realistic full-scale urban community test case, to show the accuracy and parallel performance of the proposed solver. Unless otherwise specified, all parameters of the NKS algorithm in this section are set as follows: the overlapping size $\delta = 2$ and the level of ILU fill-ins $\ell = 2$. The relative tolerances of the linear (GMRES) and nonlinear (Newton) solvers are set to $\eta_L = 1.0 \times 10^{-4}$ and $\eta_N = 1.0 \times 10^{-6}$, respectively. In the restart GMRES(k) algorithm, the restarted iteration $k = 100$. The zero initial condition is used in all the test cases.

3.1. Simulation of the flow around a single square prism

To verify our solver, a benchmark problem derived from the working group of the architectural institute of Japan (AIJ) [47] is solved, and the numerical results are compared with wind tunnel data. The working group of AIJ conducted numerous wind tunnel experiments, field measurements and computations using different CFD codes to investigate the influences of various kinds of computational parameters for various flow fields. In this benchmark problem, a 2 : 1 : 1 (height: width: depth) square prism is placed as a building in the wind tunnel, and the height and width of the model building are $h = 0.16$ m and $b = 0.08$ m, respectively. As shown in Fig. 1, the size of the computational domain is $31b \times 21b \times 12h$, the lateral boundaries are symmetrically distributed on both sides of the building model, and the distance between the inlet boundary and the building is $10b$.

The measured inlet velocity of the wind tunnel experiment and the interpolated curve used as the inlet boundary condition for the numerical simulation are shown in Fig. 1. The wind is along the x-axis, and the vertical velocity profile on the inlet surface is given by a power function of the vertical value with a power exponent of 0.27. More detailed settings can be found in [48].

We obtain the flow field on two different grids, and the numbers of tetrahedron cells are approximately 4.45×10^6 ($DOF = 3.15 \times 10^6$) and 1.25×10^7 ($DOF = 8.67 \times 10^6$), respectively. The cell sizes on the square prism walls are approximately 2 mm and 1 mm, and the averaged values of y^+ are 3.74 and 1.43, respectively. The maximum values of y^+ are near the frontal corners of the square prism. Fig. 2 shows an example of a computational mesh, where the meshes near the prism are relatively finer to resolve the complex flow structures around the prism. The literature provided the density of air at 25 °C $\rho = 1.185$ kg/m^3, the viscosity $\mu = 3.548 \times 10^{-5}$ kg/m s, and the Reynolds number $Re = 2.4 \times 10^4$. The Smagorinsky constant C_s is set to 0.1. In the numerical test, we set the inlet flow velocity linearly increase from zero to that in the experiment shown in Fig. 1 within 0.1 s. The timestep size is set to $\Delta t = 0.001$ s. The results at $t = 1.0$ s are used to compare with the experimental data. The simulation is executed on the TianHe-2A supercomputer with 720 processor cores.

Fig. 3 shows the locations of monitoring points in the wind tunnel experiment. 66 measuring points are set on the cross-section $y = 0$ m (at the center of the building) to obtain the distributions of the x and z components of wind velocity.

As shown in Fig. 4 and Fig. 5, we calculated the x component U and z component W of the wind velocity $\mathbf{u}$ at the measuring points on the vertical cross-section $y = 0$ m for comparison with the measured wind tunnel data. In these two figures, the measuring points are located at the longitudinal dotted lines, and the values of the x and z components of the wind velocity, U and W at the measuring points, are represented as the distances from the original positions. Positive values are plotted on the right side of the measuring line, with negative values on the left side.

Despite the observed differences in terms of U values between the simulation and the wind tunnel, our solver provided a similar qualitative wind flow pattern: the calculated values agree fairly well with the experimental data. Indeed, differences are found between the simulated values and experimental values, especially at the points near the ground and the top surface of the square prism.

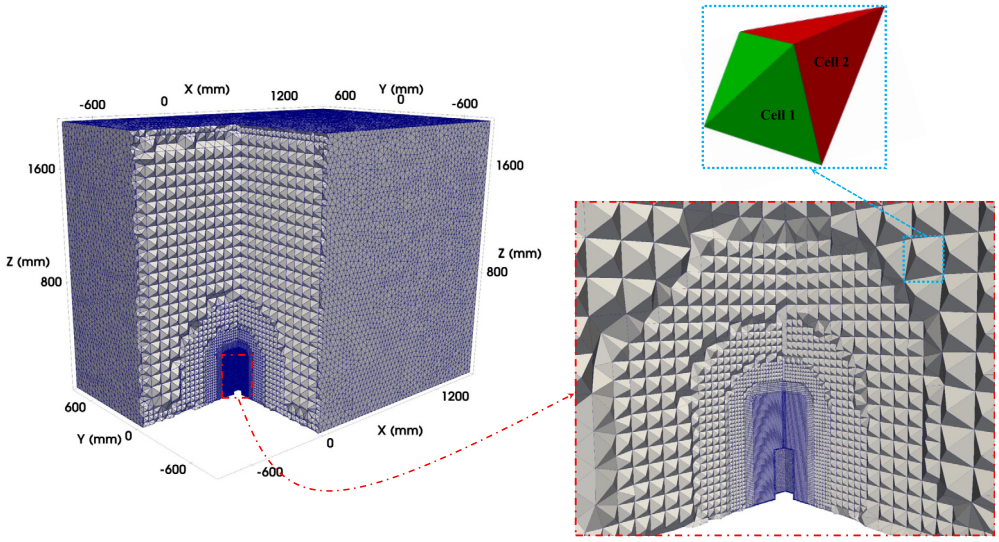


Fig. 2. Schematic view of the computational mesh in 3D (left) and a zoom-in view of the mesh near the prism (right).

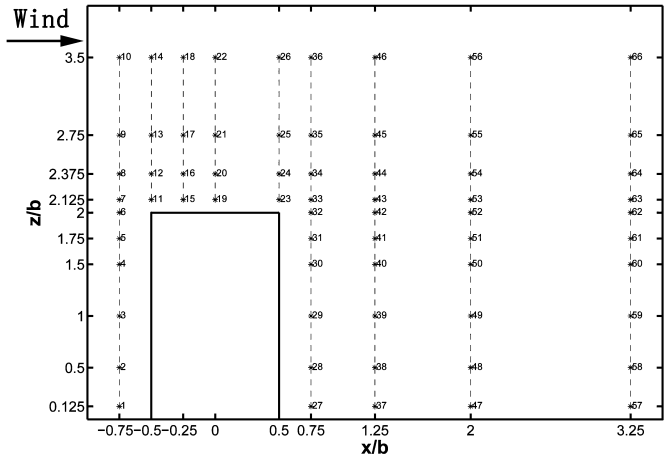


Fig. 3. Distribution of measuring points in the wind tunnel experiment [48].

At points 15, 19 and 23 near the roof surface, the simulated values of U are slightly larger than the experimental values, and the possible reason is that our fluid model does not consider the effects of a rough wall. However, we found opposite results at points 37, 47, and 57 near the ground, where the simulated values are slightly smaller than the experimental values, and the reason for this can also be attributed to the friction of the ground and the fact that the flow is reversed near these points. Similar conclusions were mentioned in previous literature [48], which also compared the CFD numerical results and the wind tunnel experimental data. The calculated results based on the two different grids are very close. The mean square percentage error (MSPE) of U at all the measuring points between the coarse grid and wind tunnel is 5.15%, where the MSPE of U is 4.99% between the fine grid and the wind tunnel. Compared with the comparison of U, the simulated vertical wind velocities W are closer to the experimental data, except for some individual points, such as points 1 and 11. The MSPEs of W based on the coarse and fine grids are 2.30% and 2.23%, respectively.

3.2. Flow simulation in a realistic full-scale urban community

In this case, we take an approximately 1 km^2 urban community, which is located in downtown Shenzhen, China, as an example of a typical downtown area of China's top-tier cities. As shown in Fig. 6, the arterial road, Shennan Avenue, passes through this area. It includes several landmarks, such as Shun Hing Square (384.0 m in height) and the KK100 mansion (441.8 m in height). The geometrical details of buildings have been proven to significantly affect the flow patterns around them, and a more detailed model could provide more information regarding the wind flow conditions of the case study area. Therefore, in this test case, the most realistic full-scale building models are used, as shown in Fig. 7. The model contains 35 high-rise buildings, and the locations of all reconstructed buildings strictly correspond to the actual locations.

The size of the computational domain is $\Omega = 4.0\ \text{km} \times 4.0\ \text{km} \times 0.8\ \text{km}$, as shown in Fig. 8. Three different meshes, named M_1, M_2 and M_3, are tested, and the numbers of tetrahedral cells are approximately $M_1 : 7.39 \times 10^6$ ($DOF = 5.03 \times 10^6$), $M_2 : 1.27 \times 10^7$ ($DOF = 9.02 \times 10^6$) and $M_3 : 2.15 \times 10^7$ ($DOF = 1.49 \times 10^7$). The maximum mesh sizes on the buildings' walls are 4.0 m, 2.5 m and 2.0 m for M_1, M_2 and M_3, respectively. Fig. 9 is a schematic view of the computational mesh.

In parallel computing, the computational domain needs to be partitioned before calculation. Fig. 10 shows an example in which the computational mesh is divided into 8 subdomains, corresponding to 8 CPU processors. It should be noted that this is just a schematic of the partition (the buildings and ground are intuitively partitioned in this example), and the actual partition should be the fluid domain. Each subdomain has a fairly equal number of elements to ensure load balancing in parallel computing. For parallel computing, the closer the number of elements in each subdomain and the fewer cut-off edges between subdomains, the better the parallel performance of the algorithm in general.

In this simulation, the physical parameters are set as follows: the air density $\rho = 1.185\ \text{kg/m}^3$, the air viscosity $\mu = 1.831 \times 10^{-5}$ kg/ms, and the Smagorinsky constant $C_s = 0.17$. The total simulation time and the timestep size are set to $T = 1800$ s and $\Delta t = 0.2$ s, respectively.

A time-dependent parabolic velocity v_{in}, which increases from zero and reaches the maximum value when $t = 1800$ s, is set as the inlet boundary condition,

$$v_{in} = -\frac{v_{max}}{1800^2}t^2 + (2\frac{v_{max}}{1800}t), \tag{8}$$

where $v_{max} = 10.0$ m/s, and the wind direction is along the positive y-axis. Taking a typical building height $H_b = 50.0$ m as the characteristic length and v_{max} as the characteristic velocity, the

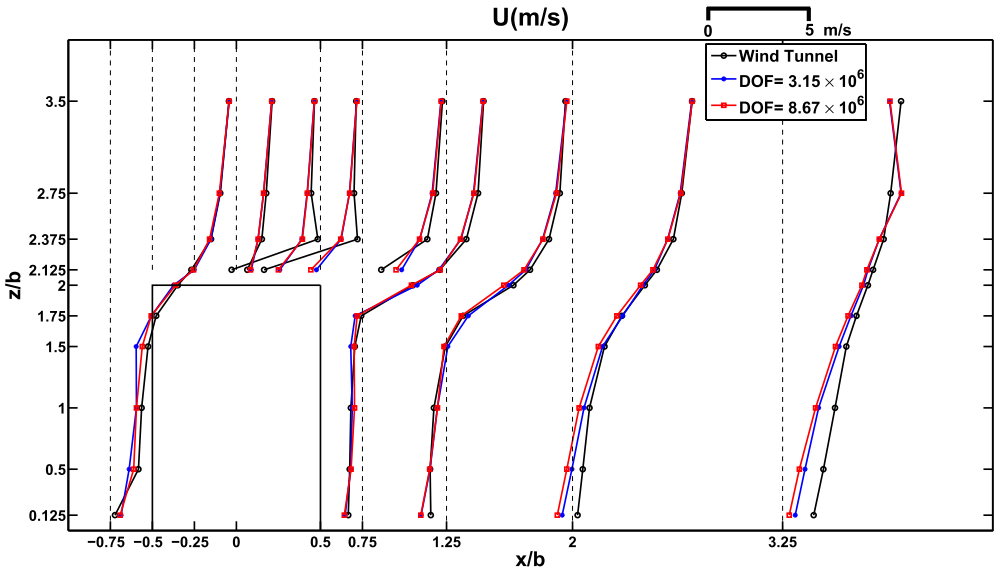


Fig. 4. Comparison of U in vertical section $y = 0$ m.

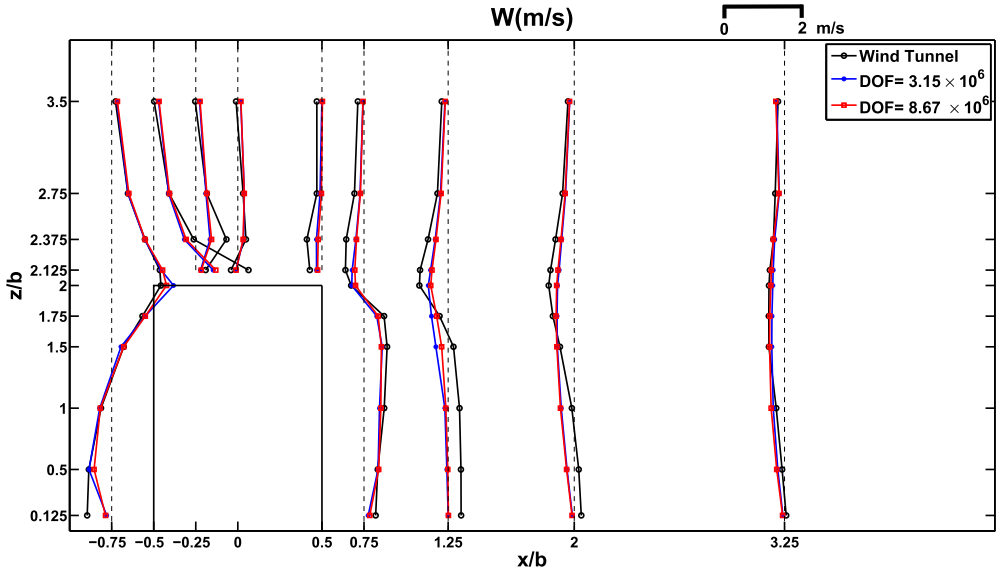


Fig. 5. Comparison of W in vertical section $y = 0$ m.

Reynolds number for this case is approximately 3.2×10^7. We assume that the air is at rest initially.

The simulations are carried out on the TianHe-2A supercomputer with 1024 CPU processors. The numerical results at $t = 1800$ s on different grids are used for showing the flow fields and comparison. Two cut planes, C_1 along Shennan Avenue and C_2 along the wind direction and tangent to the KK100 mansion, are created to show the details of the fluid motion, and a quantitative comparison of the velocity on the two lines (L_1 and L_2) on these two cut planes is also shown. The locations of the cut planes and lines are shown in Fig. 11.

Fig. 12a exhibits the velocity distribution curves on line L_1, which is along Shennan Avenue, parallel to the ground, and approximately 20 m above the ground. Due to the channeling and obstruction effects of buildings, the distribution of wind speed along the street is uneven; the maximum wind speed reaches 9.0 m/s, while the minimum wind speed is as low as 1.0 m/s. The velocity distribution curves on line L_2, which are perpendicular to the ground and behind the KK100 mansion, are shown in Fig. 12b. we can see that when the height does not exceed 441.8 m (the height of the KK100 mansion), the wind speed curves remain lower than those in Fig. 12a because of the obstruction of buildings. However, when the height reaches 441.8 m and with increasing height, the wind speed first exhibits a slight downward trend and then increases sharply, reaching a maximum value of 14.0 m/s at a height of approximately 460 m. This phenomenon can also be observed in the streamline distribution in Fig. 14, where the windward and leeward sides of the building are inclined planes, and a vortex is formed leeward of the highest point of the building, thus causing a velocity drop. At a slightly higher position over the building, with parallel winds and incoming flow from the windward side of the building, the speed reaches the maximum value. Both figures show that the results, based on the two sets of denser local grids, M_1 and M_3 are closer.

Streamlines in Fig. 13 and Fig. 14 show the flow structures on the two cross-sections, C_1 and C_2. Taking the results under the finest grid M_3 as the reference, although rather similar spatial distributions of vortices are obtained on M_1 (refine the mesh near

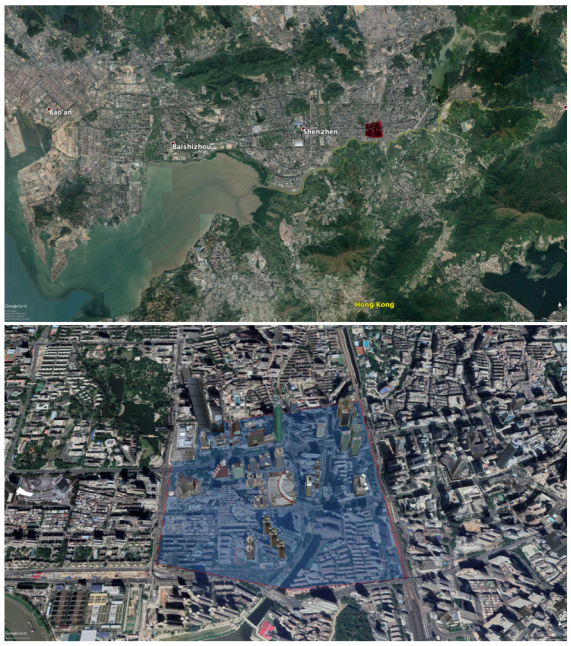


Fig. 6. Plane view of the geographic location of the tested community at Shenzhen, China.

Fig. 7. Geometry of the building Shun Hing Square that was used in the simulation.

the buildings and ground) and M_2 (refine the mesh on the building surfaces and ground), more flow details are captured on M_3, in which the meshes both on and near the buildings' surfaces and ground are refined. Especially in certain regions, such as zone A (the leeward area of the residential quarters) of the cross-section C_1 and zone B (the leeward area of the KK100 mansion) of the cross-section C_2, more small vortexes are detected. The global distribution of the vortexes under M_2 exhibits the roughest results because only the meshes on the buildings' surfaces and ground are refined, and the size of the spatial grid is too large to capture the small eddies. From the comparison, we can obtain the impression that a finer grid can provide more detailed flow field information in general, although this usually means an increase in the amount of calculation.

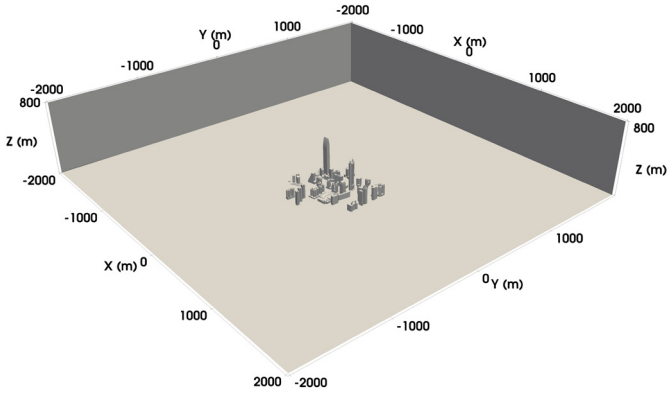


Fig. 8. Computational domain of the actual scale community model. The unit is *m*.

Fig. 15 and Fig. 16 show the wind pressure distribution around the building surfaces and on the ground. High and low pressures are observed on the windward and leeward sides of the building, respectively. The obstruction of buildings causes higher wind pressure on the windward ground. If the wind pressure is too high, such as in the case of typhoons in coastal areas, buildings, especially high-rise buildings, may experience significant damage to a particular area of the building surface, for example, glass panels. Due to the sheltering effect, low wind pressure often appears on the leeward side of buildings and forms regions of high suction. These negative pressure areas will adversely impact the spread of air pollution and form a buildup of pollutants.

Although the total number of elements of M_2 is larger than that of M_1, the local meshes near the buildings in M_2 are relatively coarser than M_1 because of the local mesh refinement in M_1. From the streamlines and pressures comparison in Fig. 13, Fig. 14 and Fig. 15, we can remark that the local grid density near the buildings affects the overall numerical results. The wake vortex structures illustrated by the Q-criterion in Fig. 17 confirm that local finer meshes can provide more detailed flow structures, such as the small vortexes, especially at the leeward side.

Fig. 18 depicts the velocity contours at a cross-section $z = 30$ m (30 m above the ground) base on M_3. Strong wind areas, in which the maximum wind speed exceeds 10.0 m/s, often appear between high-rise buildings because of the channeling effect. Wind flows will be sped up when they pass by these regions and cause damage to buildings or pedestrians. Low wind speed areas appear between dense low-rise buildings, which is also not conducive to the cleanup of air pollutants.

Due to the variety of building heights and the complexity of building shapes, the flows are very complicated, as shown in Fig. 19 and Fig. 20. The dynamics of how wind and structures interact with each other are more complex in a metropolitan center where groups of buildings are located in close proximity to one another. Various complex interference mechanisms, such as sheltering, channeling, and wake effects, can be observed from the distribution of streamlines. The details of wind motion can provide a technical foundation to better understand the nature of wind and structure interactions.

3.3. Parallel performance studies

In this section, we study the parallel performance of the proposed solver on two high-end supercomputers, TianHe-2A and Sunway TaihuLight. Three different size of grids are tested for each problem: $G_1 : DOF = 2.28 \times 10^6$, $G_2 : DOF = 4.85 \times 10^6$ and $G_3 : DOF = 1.02 \times 10^7$ for the benchmark problem; $M_1 : DOF = 5.03 \times 10^6$, $M_2 : DOF = 9.02 \times 10^6$ and $M_3 : DOF = 1.49 \times 10^7$ for the urban wind flow simulation case.

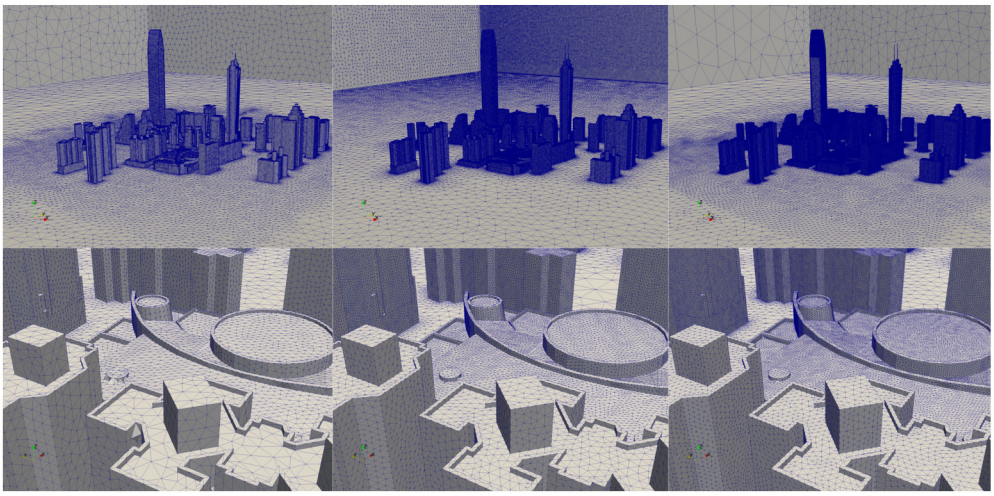


Fig. 9. Schematic view of the computational meshes. From left to right: M_1, M_2 and M_3.

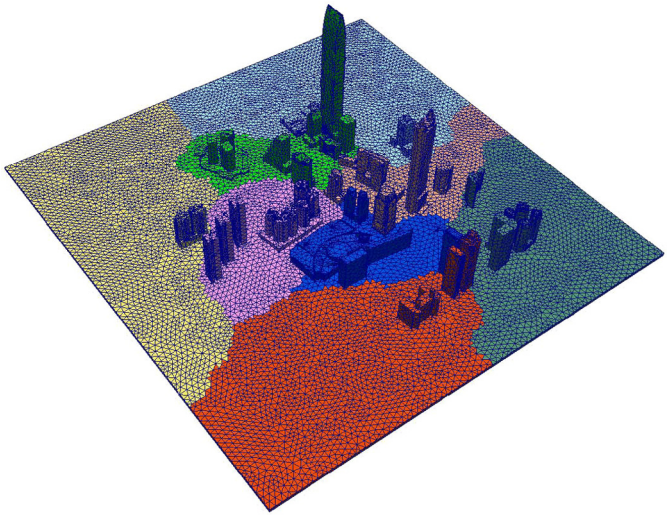


Fig. 10. Schematic view of the domain decomposition for parallel computing. Each color refers one subdomain. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

The parallel performance was first studied on TianHe-2A in both strong and weak senses. In what follows, we set the timestep size for the benchmark problem and urban wind flow simulation to be 0.001 s and 2.0 s, respectively. The relative tolerances of linear (GMRES) and nonlinear (Newton) solvers are set to $\eta_L = 1.0 \times 10^{-8}$ and $\eta_N = 1.0 \times 10^{-12}$, respectively. In all tables and figures, "n_p" denotes the number of processors, and "NI" and "LI" are the average number of nonlinear iterations per timestep and the average number of GMRES iterations per Newton step, respectively. "Time" is the average total computation time in seconds per timestep, "Ideal" is the ideal speedup, and "Eff." is the parallel efficiency. To save the computing cost and remove the impact of the initial timestep, the results are analyzed based on the second 20 timesteps.

The strong scalability results for the benchmark problem are presented in Table 1 and Fig. 21. Compared with the result of $n_p = 256$, the proposed solver achieves over 50% parallel efficiency when the number of processors is up to 4,096. The parallel efficiency of G_3 is higher than that of G_1 and G_2 with the same number of processors. This is true because the computation time includes not only the subdomain solution time but also the communication time among processors which is not parallel scalable.

Table 1
Strong scalability of the proposed solver for the benchmark case carried out on Tianhe-2A. Here, the subdomain solver is the ILU factorization method with 2 levels of fill-in.

n_p		256	512	1024	2048	4096
$G_1 : DOF = 2.28 \times 10^6$	Time (s)	20.6	11.7	7.0	4.9	3.4
	NI	2.7	2.7	2.7	2.8	2.8
	LI	47.5	49.3	51.1	55.8	60.8
	Eff.	100.0%	88.5%	74.1%	52.6%	37.6%
$G_2 : DOF = 4.85 \times 10^6$	Time (s)	45.5	24.1	14.1	8.9	5.9
	NI	2.8	2.6	2.5	2.5	2.5
	LI	59.0	60.7	66.2	70.1	75.5
	Eff.	100.0%	94.5%	80.7%	63.7%	48.3%
$G_3 : DOF = 1.02 \times 10^7$	Time (s)	98.1	56.7	32.0	19.1	12.8
	NI	2.8	3.0	3.0	3.0	3.0
	LI	69.2	74.4	79.2	84.4	90.7
	Eff.	100.0%	87.2%	78.0%	64.1%	50.2%

Table 2
Strong scalability of the proposed solver for the urban wind flow simulation carried out on Tianhe-2A. Here, the subdomain solver is the ILU factorization method with 2 levels of fill-in.

n_p		1024	2048	4096	8192
$M_1 : DOF = 5.03 \times 10^6$	Time (s)	26.1	16.0	11.1	9.5
	NI	3.6	3.6	3.6	3.6
	LI	145.5	152.6	167.0	209.8
	Eff.	100.0%	82.0%	59.0%	34.7%
$M_2 : DOF = 9.02 \times 10^6$	Time (s)	53.8	33.6	23.6	20.2
	NI	4.0	4.0	4.0	4.0
	LI	164.7	166.9	175.4	200.9
	Eff.	100.0%	80.2%	57.1%	33.3%
$M_3 : DOF = 1.49 \times 10^7$	Time (s)	98.8	60.0	37.5	28.9
	NI	3.1	3.1	3.1	3.1
	LI	318.4	332.0	339.1	380.1
	Eff.	100.0%	82.8%	65.9%	42.7%

The size of the subdomain problem of G_3 is larger than that of G_1 and G_2, which means that there is a smaller proportion of communication time among the total computation time, and it therefore displays a higher parallel efficiency.

Table 2 and Fig. 22 present the strong scaling results for the urban wind flow simulation. Similar to the benchmark problem, the efficiency of the solver increases with the increase of DOF and it achieves 42.7% for the finest mesh M_3 with up to 8,192 processors. Besides the execution time, the number of Newton iterations and the average number of GMRES iterations with respect to the

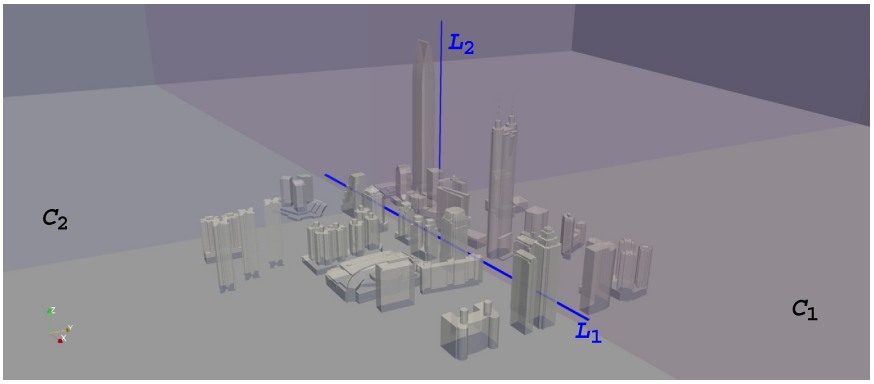
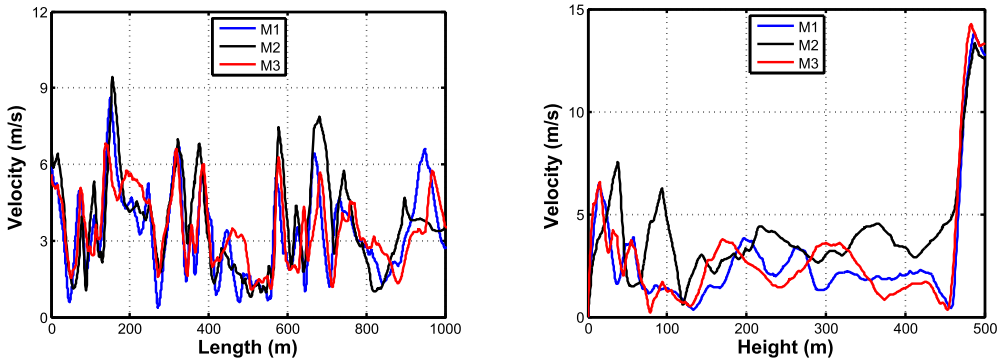


Fig. 11. The locations of monitoring points, cut planes and lines.



(a) Line L_1 (b) Line L_2

Fig. 12. Velocity magnitude on lines L_1 and L_2 under different grids at $t = 1800$ s.

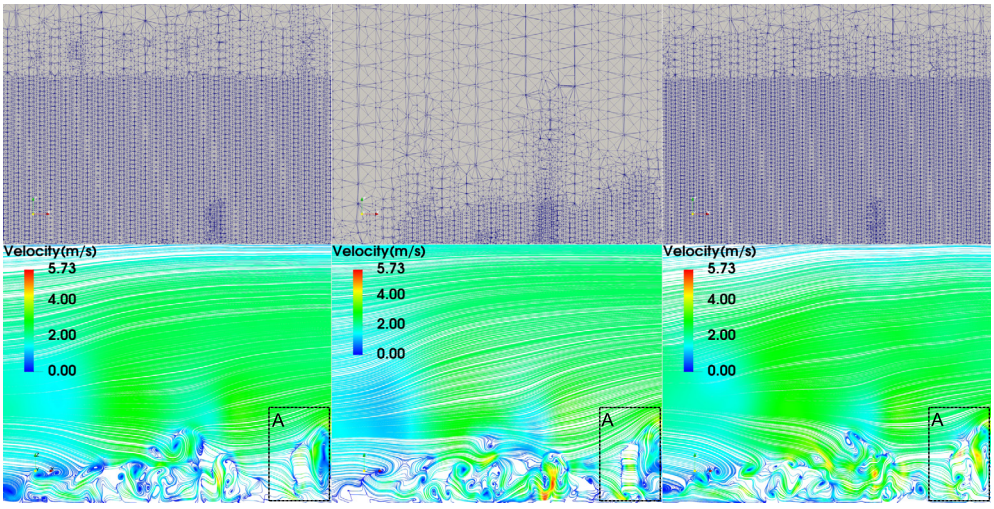


Fig. 13. Computational meshes (top) and streamlines (bottom) on cross-section C_1 under different grids at $t = 1800$ s. From left to right: M_1, M_2 and M_3.

number of processors and the problem sizes are also summarized in Table 1 and Table 2. We observed that the number of GMRES iterations increase slowly as n_p increases. The number of Newton iterations is always kept at a low level with the increase of n_p while the size of the overall mesh is fixed, which indicates that the solver is nonlinearly scalable.

Base on the results on different grids listed in Table 1 and Table 2, we plot the weak scalability of the solver for the benchmark problem and the wind flow simulation in Fig. 23. Since the DOFs of the grids in each comparison group are not strictly proportional, the weak scaling efficiency E_w on grid k is defined as: $E_w^k = (T^{ref} \times DOF^k \times {n_p}^k)/(T^k \times DOF^{ref} \times {n_p}^{ref})$, where T^{ref}, DOF^{ref} and ${n_p}^{ref}$ are the compute time, DOF and n_p of the reference grid (the coarsest grid) in each group, and T^k, DOF^k and ${n_p}^k$ are those on grid k. From Fig. 23 we observed that the weak scaling efficiency of the proposed solver achieves 63.7% for the benchmark problem with up to 4,096 processors and 40.9% for the urban wind flow case with up to 8,192 processors.

We also tested the solver's parallel performance on the Sunway TaihuLight supercomputer using the urban wind flow case. As we mentioned in subsection 2.3, the NKS method contains many parameters that may affect the solver's parallel performance.

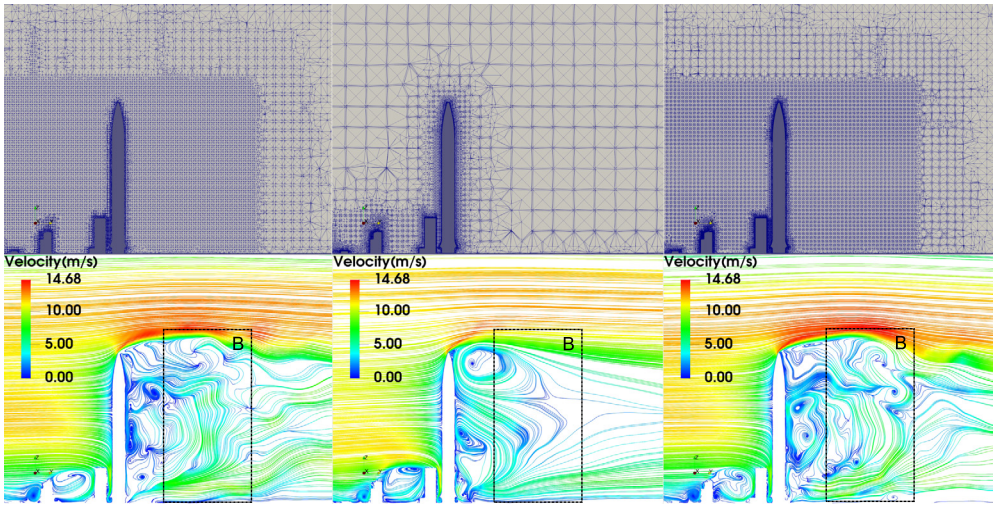


Fig. 14. Computational meshes (top) and streamlines (bottom) on cross-section C_2 under different grids at $t = 1800$ s. From left to right: M_1, M_2 and M_3.

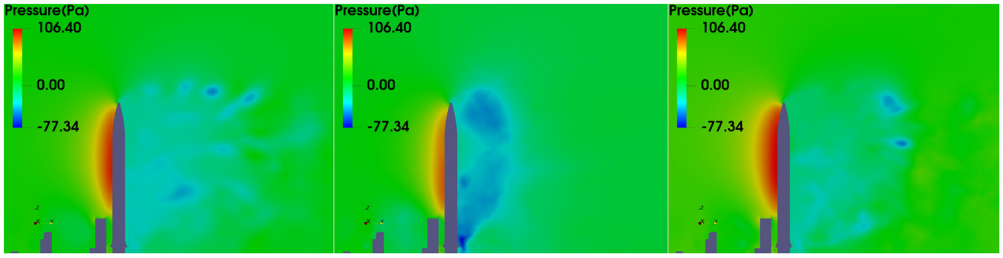


Fig. 15. Pressure contours on cross-section C_1 under different grids at $t = 1800$ s. From left to right: M_1, M_2 and M_3.

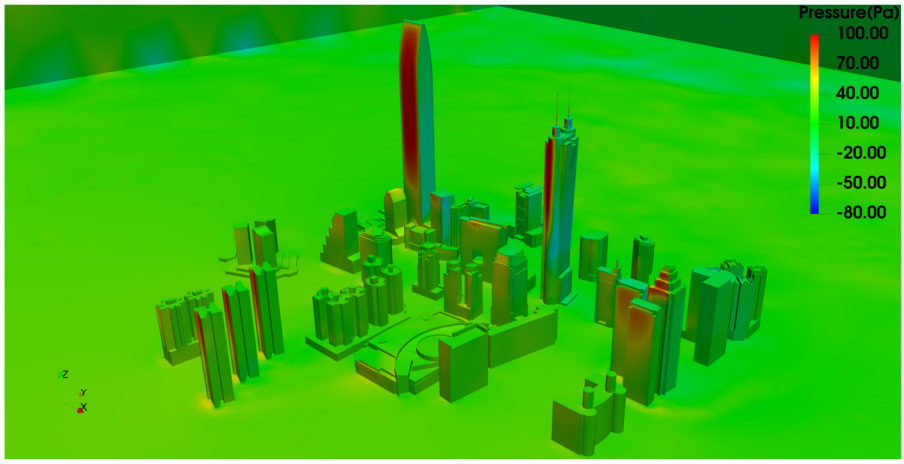


Fig. 16. The wind pressure distribution on building surfaces and ground obtained on M_3.

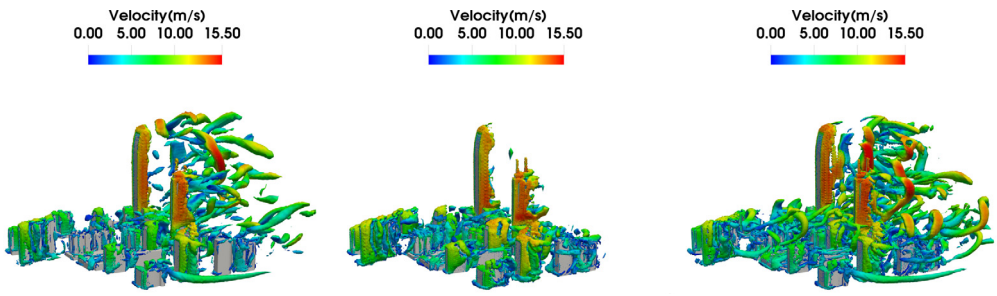


Fig. 17. Vortex structures around the buildings illustrated by the Q-criterion at $t = 1800$ s. Colored by the velocity (Q = 0.02). From left to right: M_1, M_2 and M_3.

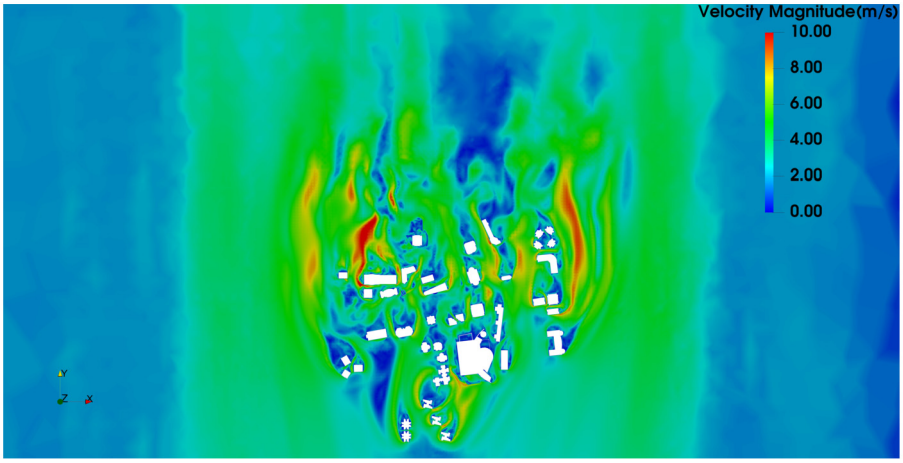


Fig. 18. Velocity magnitude contours at cross-section $z = 30$ m, obtained on M_3.

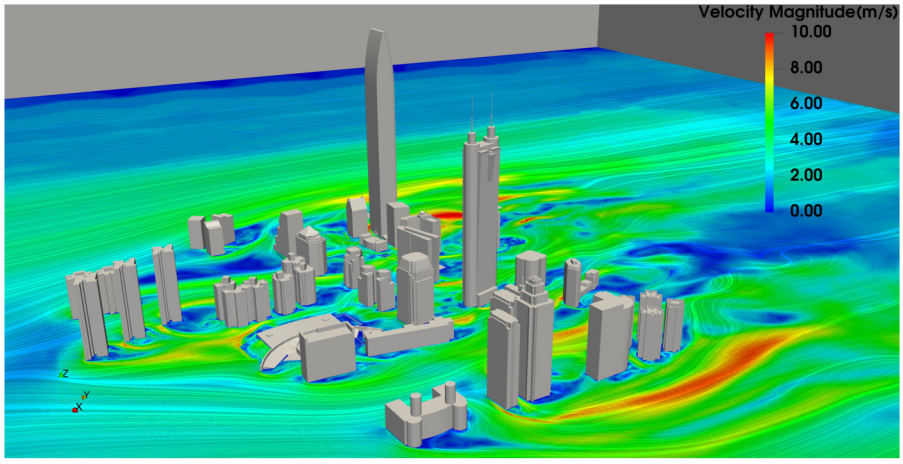


Fig. 19. Surface streamlines at cross-section $z = 30$ m obtained on the mesh M_3.

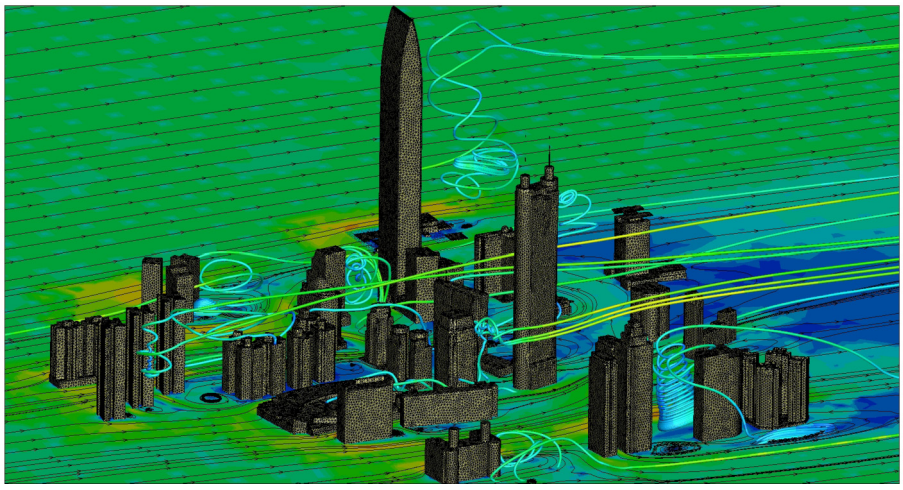


Fig. 20. The 3D streamlines distribution around the buildings obtained on the mesh M_3.

Here, we show how different subdomain solvers, LU and ILU factorization, impact the computation time and the parallel performance.

Table 3 shows the scalability of the solver on the Sunway TaihuLight supercomputer. An restricted additive Schwarz preconditioned GMRES with LU as a subdomain solver is more scalable than that with ILU. Although LU factorization can achieve a superlinear parallel speedup, the computing cost is significantly high, especially when the number of processors is less than 4,096. Fig. 24 presents a more intuitive view of the solver's performance in the form of computation time and speedup. When n_p reaches 8,192, the LU-based efficiency is 40.8%, which is slightly lower than that on TianHe-2A. When further increasing n_p up to 16, 384, the solver only performs with an efficiency of 27.4%. The reason for the fast

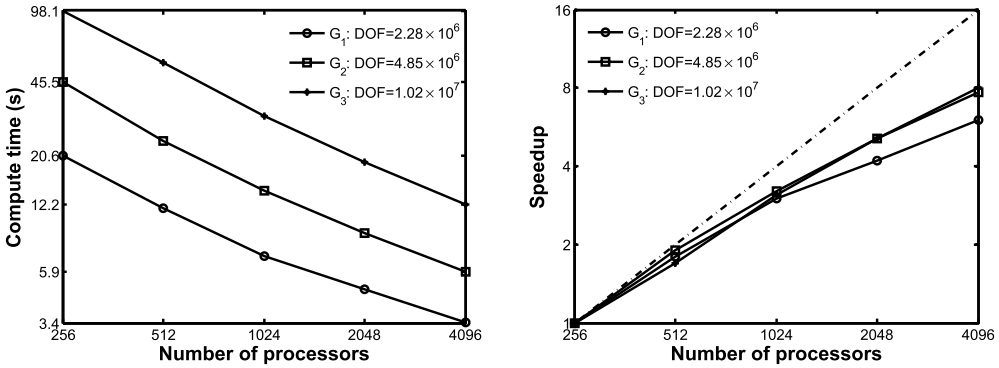


Fig. 21. The average total computation time per timestep (left) and the speedup (right) for the benchmark problem carried out on Tianhe-2A.

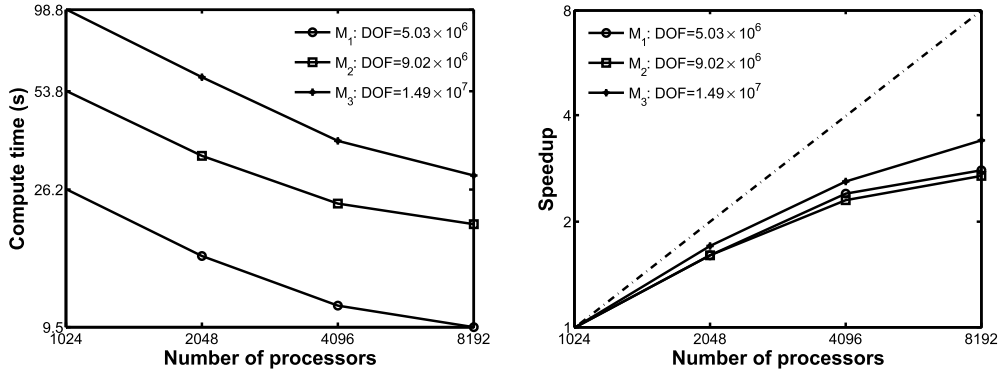


Fig. 22. The average total computation time per timestep (left) and the speedup (right) for the urban wind flow simulation carried out on Tianhe-2A.

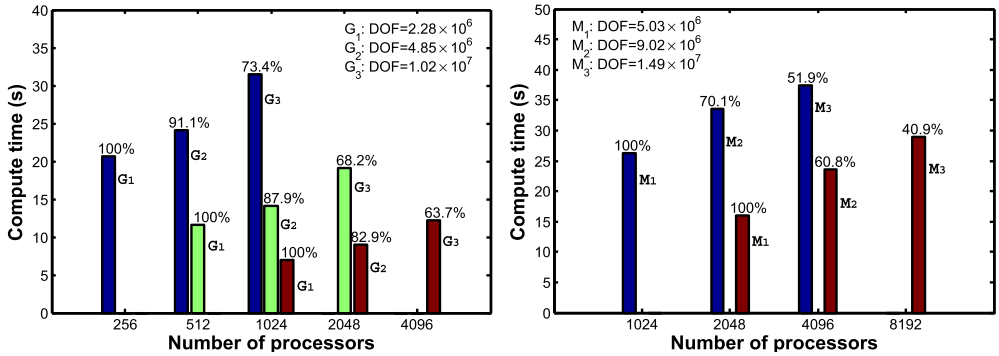


Fig. 23. Weak scalability for the benchmark problem (left) and the wind flow simulation (right), carried out on Tianhe-2A. Results are color coded according to group they belong. The percentages are the corresponding weak scaling efficiencies.

Table 3
Impact of the subdomain solvers on the parallel performance of the proposed solver. Here the tested grid is $M_3 : DOF = 1.49 \times 10^7$ and the simulation is carried out on Sunway TaihuLight.

n_p	LU				ILU			
	Time (s)	Speedup	Ideal	Eff.	Time (s)	Speedup	Ideal	Eff.
1024	1051.7	1	1	100.0%	139.5	1	1	100.0%
2048	351.2	3.00	2	149.7%	93.2	1.50	2	74.8%
4096	117.3	8.97	4	224.2%	66.1	2.11	4	52.8%
8192	60.1	17.50	8	218.7%	42.7	3.27	8	40.8%
16384	31.8	27.97	16	174.8%	31.8	4.39	16	27.4%

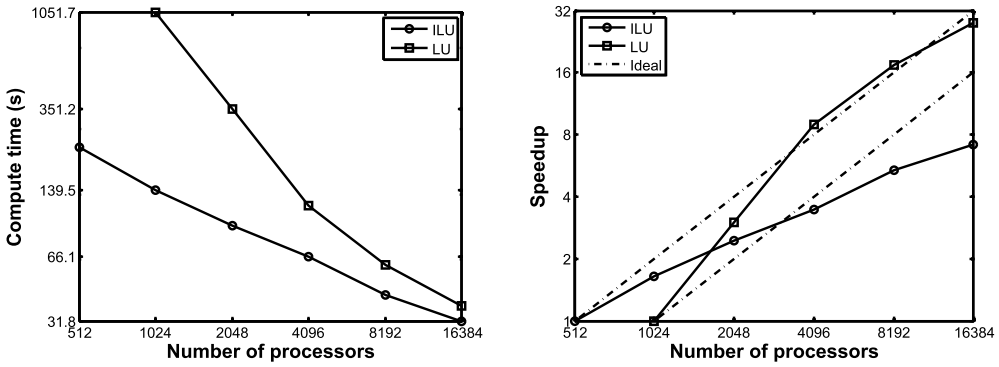


Fig. 24. The average total computation time per timestep and the speedup (carried out on Sunway TaihuLight, based on $M_3 : DOF = 1.49 \times 10^7$).

drop in efficiency may be related to the features of the master-slave multithread heterogeneous architecture of the supercomputer. In Sunway TaihuLight, each processor has four computing groups, each of which has one management processing element (master core) and a 8×8 array of computing processing elements (slave cores), resulting in a total of 260 cores. We only utilized the master cores in this computing; thus, the overall performance was greatly affected, and the solver should be further optimized for this architecture to improve the parallel performance.

4. Conclusions

Large urban areas, high resolution, and realistic building models as well as high Reynolds number, make the large eddy simulation of the urban wind flow a challenging problem. A primary concern with solvers is the parallel scalability, which indicates the ability to deliver more computational power when the amount of resources is increased. Based on a parallel Newton-Krylov-Schwarz method, this paper studies the large eddy simulation of the flows around a benchmark problem, as well as a realistic and full-scale urban community. Although there are some deviations from the wind tunnel data, our numerical results match the flow field well for the benchmark case. The application of urban community wind simulation shows that typical complex flow phenomena, such as building sheltering, channeling, and wake effects, can be effectively caught. The solver presents 42.7% parallel efficiency when the number of processors is up to $8,192$ on the TianHe-2A supercomputer, along with good scalability. The solver scales up to $16,384$ processors on the Sunway TaihuLight supercomputer with 27.4% parallel efficiency. These results show that the proposed solver has the potential to perform fast and high-fidelity simulations of large-scale wind flows in complicated computational domains.

The current version of the solver can only simulate the flow in a small urban community with thousands of processors because of the scalability issue. To perform a city-wide simulation with tens of thousands of processors, more advanced solvers are needed. In the future, we plan to study some multilevel Schwarz methods that may improve the scalability when the number of processors is large.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We acknowledge the National Supercomputer Center in Guangzhou and Wuxi, China for the valuable comments that helped to implement algorithms on the TianHe-2A and Sunway TaihuLight supercomputers. This work was partially supported by the National Key R & D Program [No. 2017YFB0202103]; the NSF of China [No. 11901559, 12071461]; and the Shenzhen grants [No. ZDSYS201703031711426 and JCYJ20180507182506416].

References

[1] A. Dagnew, G.T. Bitsuamlak, Wind Struct. 16 (6) (2013) 629–660.
[2] C. Gromke, B. Ruck, Bound.-Layer Meteorol. 144 (1) (2012) 41–64.
[3] Z. Mo, C.-H. Liu, Build. Environ. 132 (2018) 357–366.
[4] B. Blocken, W. Janssen, T. van Hooff, Environ. Model. Softw. 30 (2012) 15–34.
[5] Y. Toparlar, B. Blocken, B. Maiheu, G. Van Heijst, Renew. Sustain. Energy Rev. 80 (2017) 1613–1640.
[6] T. van Hooff, B. Blocken, Y. Tominaga, Build. Environ. 114 (2017) 148–165.
[7] B. Blocken, T. Stathopoulos, J. Van Beeck, Build. Environ. 100 (2016) 50–81.
[8] M. Roth, Q. J. R. Meteorol. Soc. 126 (564) (2000) 941–990.
[9] M. Cao, Z. Lin, J. Appl. Math. 2014 (2014).
[10] Z. Peng, J. Sun, Bound.-Layer Meteorol. 153 (3) (2014) 569–580.
[11] R.D. Crago, W. Okello, M.F. Jasinski, Bound.-Layer Meteorol. 145 (3) (2012) 423–437.
[12] H. Montazeri, B. Blocken, Build. Environ. 60 (2013) 137–149.
[13] Y. Tominaga, T. Stathopoulos, Atmos. Environ. 43 (20) (2009) 3200–3210.
[14] B. Blocken, T. Stathopoulos, J. Carmeliet, J. Hensen, in: Eleventh International IBPSA Conference, vol. 4, 2009, pp. 157–184.
[15] Y. Tominaga, T. Stathopoulos, Atmos. Environ. 79 (2013) 716–730.
[16] Z. Ai, C. Mak, Comput. Fluids 118 (2015) 89–100.
[17] P. Gousseau, B. Blocken, T. Stathopoulos, G.F. van Heijst, Comput. Fluids 114 (2015) 151–162.
[18] J. Liu, M. Heidarinejad, G. Pitchurov, L. Zhang, J. Srebric, Sustain. Cities Soc. 40 (2018) 28–43.
[19] P.R. Spalart, in: Proceedings of First AFOSR International Conference on DNS/LES, Greyden Press, 1997.
[20] J. Liu, J. Niu, Build. Environ. 96 (2016) 91–106.
[21] K. Nakajima, R. Ooka, H. Kikumoto, J. Wind Eng. Ind. Aerodyn. 175 (2018) 213–228.
[22] F. Toja-Silva, T. Kono, C. Peralta, O. Lopez-Garcia, J. Chen, J. Wind Eng. Ind. Aerodyn. 180 (2018) 66–87.
[23] S. Liu, W. Pan, X. Zhao, H. Zhang, X. Cheng, Z. Long, Q. Chen, Build. Environ. 140 (2018) 1–10.
[24] I.C. Tolias, N. Koutsourakis, D. Hertwig, G.C. Efthimiou, A.G. Venetsanos, J.G. Bartzis, J. Wind Eng. Ind. Aerodyn. 177 (2018) 101–116.
[25] B. Maronga, M. Gryschka, R. Heinze, F. Hoffmann, F. Kanani-Sühring, M. Keck, K. Ketelsen, M.O. Letzel, M. Sühring, S. Raasch, Geosci. Model Dev. 8 (8) (2015) 2515–2551.
[26] M.O. Letzel, C. Helmke, E. Ng, X. An, A. Lai, S. Raasch, Meteorol. Z. 21 (6) (2012) 575–589.
[27] M. Kurppa, A. Hellsten, M. Auvinen, S. Raasch, T. Vesala, L. Järvi, Atmosphere 9 (2) (2018) 65.
[28] M. Auvinen, S. Boi, A. Hellsten, T. Tanhuanpää, L. Järvi, Atmosphere 11 (2) (2020) 201.
[29] N. Onodera, T. Aoki, T. Shimokawabe, H. Kobayashi, J. Glob. Sci. Inf. Comput. Cent. 9 (2013) 1–8.
[30] N.H. Ahmad, A. Inagaki, M. Kanda, N. Onodera, T. Aoki, Bound.-Layer Meteorol. 163 (3) (2017) 447–467.
[31] S. Lenz, M. Schünherr, M. Geier, M. Krafczyk, A. Pasquali, A. Christen, M. Giometto, J. Wind Eng. Ind. Aerodyn. 189 (2019) 151–162.
[32] J.R. Sack, J. Urrutia, Handbook of Computational Geometry, Elsevier, 1999.
[33] J. Smagorinsky, Mon. Weather Rev. 91 (3) (1963) 99–164.

[34] L.P. Franca, S.L. Frey, Comput. Methods Appl. Mech. Eng. 99 (2–3) (1992) 209–233.
[35] X.-C. Cai, W.D. Gropp, D.E. Keyes, M.D. Tidriri, in: Numerical Methods for the Navier-Stokes Equations, Springer, 1994, pp. 17–30.
[36] S.C. Eisenstat, H.F. Walker, SIAM J. Optim. 4 (2) (1994) 393–422.
[37] Y. Saad, M.H. Schultz, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.
[38] X.-C. Cai, M. Sarkis, SIAM J. Sci. Comput. 21 (2) (1999) 792–797.
[39] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.
[40] S.C. Eisenstat, H.F. Walker, SIAM J. Sci. Comput. 17 (1) (1996) 16–32.
[41] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Karpeyev, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Web page, https://petsc.org, 2021.
[42] ANSYS Inc., ICEM CFD User's manual, release 19.2, 2018.
[43] G. Karypis, V. Kumar, ParMETIS–parallel graph partitioning and fill-reducing matrix ordering, version 4, http://www.cs.umn.edu/~metis, 2014.
[44] U. Ayachit, The ParaView Guide: A Parallel Visualization Application, Kitware, Inc., Clifton Park, NY, USA, 2015.
[45] Z. Liao, R. Chen, Z. Yan, X.-C. Cai, Int. J. Numer. Methods Fluids 89 (2019) 343–361.
[46] R. Chen, B. Wu, Z. Cheng, W. Shiu, J. Liu, L. liu, Y. Wang, X. Wang, X.-C. Cai, Int. J. Numer. Methods Biomed. Eng. 36 (2020) e33392.
[47] A. Mochida, Y. Tominaga, S. Murakami, R. Yoshie, T. Ishihara, R. Ooka, Wind Struct. 5 (2–4) (2002) 227–244.
[48] R. Yoshie, A. Mochida, Y. Tominaga, H. Kataoka, K. Harimoto, T. Nozu, T. Shirasawa, J. Wind Eng. Ind. Aerodyn. 95 (9–11) (2007) 1551–1578.