


RESEARCH ARTICLE

A parallel implicit domain decomposition algorithm for the large eddy simulation of incompressible turbulent flows on 3D unstructured meshes

Zi-Ju Liao¹ | Rongliang Chen^{2,3} | Zhengzheng Yan² | Xiao-Chuan Cai⁴ 

¹Department of Mathematics, College of Information Science and Technology, Jinan University, Guangzhou, China

²Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

³Shenzhen Key Laboratory for Exascale Engineering and Scientific Computing, Shenzhen, China

⁴Department of Computer Science, University of Colorado Boulder, Boulder, Colorado

Correspondence

Xiao-Chuan Cai, Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309.
Email: cai@cs.colorado.edu

Funding information

National Key R&D Program of China, Grant/Award Number: 2016YFB0200601; Shenzhen Basic Research Program, Grant/Award Number: JCYJ20160331193229720, JCYJ20170307165328836, and JSGG20170824154458183; National Natural Science Foundation of China, Grant/Award Number: 61531166003, 11602282, and DMS-1720366

Summary

We present a parallel fully implicit algorithm for the large eddy simulation (LES) of incompressible turbulent flows on unstructured meshes in three dimensions. The LES governing equations are discretized by a stabilized Galerkin finite element method in space and an implicit second-order backward differentiation scheme in time. To efficiently solve the resulting large nonlinear systems, we present a highly parallel Newton-Krylov-Schwarz algorithm based on domain decomposition techniques. Analytic Jacobian is applied in order to obtain the best achievable performance. Two benchmark problems of lid-driven cavity and flow passing a square cylinder are employed to validate the proposed algorithm. We then apply the algorithm to the LES of turbulent flows passing a full-size high-speed train with realistic geometry and operating conditions. The numerical results show that the algorithm is both accurate and efficient and exhibits a good scalability and parallel efficiency with tens of millions of degrees of freedom on a computer with up to 4096 processors. To understand the numerical behavior of the proposed fully implicit scheme, we study several important issues, including the choices of linear solvers, the overlapping size of the subdomains, and, especially, the accuracy of the Jacobian matrix. The results show that an exact Jacobian is necessary for the efficiency and the robustness of the proposed LES solver.

KEYWORDS

domain decomposition, large eddy simulation, Newton-Krylov-Schwarz, parallel computation, unstructured mesh

1 | INTRODUCTION

We develop a highly parallel algorithm for the large eddy simulation (LES) of incompressible turbulent flows on unstructured meshes in three dimensions (3D). LES is a technique that is intermediate between the direct numerical simulation (DNS) and the Reynolds-averaged Navier-Stokes (RANS) simulation. In LES, only the large scales of the flow field are fully represented and resolved, and the effect of the unresolved small scales of turbulence is modeled. LES can provide reliable solutions for complex flows at a relatively lower cost than DNS; and the turbulence models of LES are usually simpler and require fewer adjustments than those of RANS when applied to different flows, since the small scales in the flow tend to

be more homogeneous and isotropic. However, it is still a major challenge in applying LES to complex flow problems in engineering applications due to the tremendous computational resources required. As the rapid advancement of supercomputers, a feasible way to solve this problem is to develop highly scalable parallel algorithms for LES. In this paper, we introduce a fully implicit, highly parallel LES algorithm on unstructured meshes for incompressible turbulent flows and test it for flows around a high-speed train with a realistic geometry and high Reynolds number.

There are many publications on parallel algorithms for the computational fluid dynamics (CFD). We refer in the following some of recent works related to LES. Gokarn et al¹ presented a parallel finite difference scheme to solve the incompressible LES equations, in which the pressure Poisson equation is solved by a biconjugate gradient method with stabilization and least-squares preconditioning. Hsu et al² developed a finite volume method for LES and a multilevel Schwarz preconditioned conjugate gradient method for the pressure Poisson equation. A convection-stabilized mixed finite element scheme was studied by Colomés and Badia,³ in which the discretization system is solved by a highly scalable balancing domain decomposition by constraints method. Their approach scales well with up to 8000 processors for structured meshes. Situ et al⁴ showed a 3D LES implementation using a nonoverlapping truncated SPIKE algorithm and achieved a strong-scaling parallel efficiency of 74% at 91 125 processors with respect to a baseline of 2744 processors. Singh et al⁵ provided an assessment of different parallel preconditioners for numerical solution of the pressure Poisson equation arising in LES of turbulent incompressible flows. The aforementioned methods are all based on structured grids that are dominant in parallel LES computations; however, the demand is high for unstructured grid LES solvers in engineering applications defined on irregular domains. We now briefly recall some of the recent works on unstructured grid methods. Antepara et al⁶ proposed a parallel adaptive mesh refinement strategy for LESs that achieves a parallel efficiency of 90% on 256 CPU-cores. Su and Yu⁷ studied a parallel finite volume method for the LES of turbulent flows around the side mirror of a car. Forti and Dedè⁸ used a semi-implicit time discretization for the Navier-Stokes equations with a variational multiscale-large eddy simulation (VMS-LES) model for the turbulence, in which a good scalability with up to 4096 processors was reported. Moureau et al⁹ developed an incompressible finite volume solver called “Yales2” that scales well with up to 32 768 processors. Nichols et al¹⁰ applied a CharLES finite volume method to simulate a turbulent jet flow with hundreds of millions of grid points and with up to 163 840 processor cores. It is important to note that the temporal discretization of most of the existing approaches are explicit or semi-implicit, and such methods do not require a nonlinear solver at each time step but have a severe restriction on the time step size due to the Courant-Friedrichs-Lewy (CFL) stability condition. Implicit methods are more stable and allow a much larger time step size that depends only on the accuracy requirement. However, most research and commercial CFD software packages that use implicit schemes are only able to scale up to a small number of processors,¹¹ which is too slow if used in the inner loop within a design cycle.

In this work, we introduce a fully implicit finite element method for the LES of incompressible turbulent flows on 3D unstructured meshes. We focus on the robustness and the parallel scalability, both are very important for realistic engineering applications in production mode. We employ the Smagorinsky model to close the LES equations. Then, the LES equations are discretized by a $P_1 - P_1$ stabilized finite element method in space and an implicit backward differentiation formulas (BDFs) of order 2 in time. We use an NKS algorithm to solve the resulting large nonlinear algebraic system of equations. The NKS algorithm consists of three major components: an inexact Newton method to solve the nonlinear systems, a Krylov subspace-type method to solve the linear Jacobian system at each Newton step, and an overlapping Schwarz-type preconditioner to accelerate the convergence.

NKS has been successfully applied to solve different kind of nonlinear problems, for example, PDE-constrained optimization problems,¹² fluid-structure interaction problems,^{13,14} non-Newtonian fluid problems,¹⁵ inverse source problems,¹⁶ and elasticity problems,¹⁷ and has shown good parallel scalability to thousands of processors. In this work, we extend the algorithm to solve the fully implicit 3D LES problems and to investigate the performance of NKS for an industrial application.

Note that there are also other approaches to handle the nonlinear systems, for example, linearize the nonlinear terms by extrapolating the solution from the previous time steps based on Newton-Gregory backward polynomials.⁸ These treatments work well in most cases but may not be stable when the time step size is large. In NKS, a critically important step is the calculation of the Jacobian matrix, which can be obtained analytically or approximately. Our experiments show that, a more accurate Jacobian improves the convergence rate and the robustness of the Newton method and can also provide a better preconditioner for the linear solver. Therefore, in our study, we compute the Jacobian matrix analytically by hand even though it is rather a sophisticated task (especially for the convective nonlinear and the subgrid model terms). To evaluate the effect of the accuracy of the Jacobian matrix on the performance of the NKS algorithm, we include a study of the convergence behavior of NKS with approximately computed Jacobian.

As test cases, we first consider the benchmark lid-driven cavity flow at Reynolds number 10 000 as well as the square cylinder flow problem at Reynolds number 22 000. These are typical cases for internal flows and external flows, respectively, and they cover the main features of turbulent flows such as flow separation, vortex shedding, and secondary flows. We validate the proposed method against other published LES computations and experiments. We then apply the method to simulate flows around a full-size high-speed train with eight cars. The realistic geometry of the train and the realistic operation speed 360 km/h are adopted in our simulation. The flow is very complex and turbulent due to the blunt-slender body configuration and the high Reynolds number (2.14×10^7). It is a very challenging problem in CFD, and to the best of our knowledge, it is the first trial to conduct such LES calculation for a realistic high-speed train. We use this case to investigate the numerical behavior and the parallel performance and scalability of the proposed algorithm.

The remainder of this paper is organized as follows. In Section 2, we describe the governing equations, the subgrid-scale (SGS) models for LES, the finite element discretization of the flow problem, and the NKS algorithm for solving the discretized system of equations. In Section 3, we provide some numerical results and report the parallel performance of the proposed algorithm. Some concluding remarks are given in Section 4.

2 | NUMERICAL METHODS

2.1 | Governing equations

We consider LES of the incompressible turbulent flows. In LES, only the large scale motions are computed directly, and hence, a low-pass spatial filter is applied to the instantaneous flow fields (denoted by an overbar). The governing equations of LES are the filtered incompressible Navier-Stokes equations in a bounded domain $\Omega \subset R^3$

$$\begin{cases} \frac{\partial \bar{\mathbf{u}}}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}} = -\frac{1}{\rho} \nabla \bar{p} + \nabla \cdot (2\nu \bar{\mathbf{S}}) - \nabla \cdot \boldsymbol{\tau}, \\ \nabla \cdot \bar{\mathbf{u}} = 0, \end{cases} \quad (1)$$

where ρ is the fluid density, ν is the kinematic viscosity, $\bar{\mathbf{u}}(\mathbf{x}, t)$ is the filtered velocity field, $\bar{p}(\mathbf{x}, t)$ is the filtered pressure field, $\bar{\mathbf{S}}$ is the filtered strain rate tensor defined as

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

and $\boldsymbol{\tau}$ denotes the SGS stress tensor given by

$$\boldsymbol{\tau} = \overline{\mathbf{u}\mathbf{u}} - \bar{\mathbf{u}}\bar{\mathbf{u}}. \quad (3)$$

The SGS stress accounts for the effects of the small unresolved scales on the dynamics of the large resolved scales and has to be modeled in order to close the system. In the present study, both the Smagorinsky and dynamic Smagorinsky models are incorporated into the system.

In the Smagorinsky model,¹⁸ $\boldsymbol{\tau}$ is modeled by introducing an eddy-viscosity concept such that

$$\boldsymbol{\tau}^d \equiv \boldsymbol{\tau} - \left(\frac{1}{3} \text{tr } \boldsymbol{\tau}\right) \mathbf{I} = -2\nu_t \bar{\mathbf{S}}, \quad (4)$$

where $\boldsymbol{\tau}^d$ is the deviatoric part of the SGS stress, and ν_t is the turbulent eddy viscosity given by

$$\nu_t = (C_s \bar{\Delta})^2 |\bar{\mathbf{S}}|. \quad (5)$$

Here, $|\bar{\mathbf{S}}| = (2\bar{\mathbf{S}} : \bar{\mathbf{S}})^{1/2} = (2\bar{S}_{ij}\bar{S}_{ij})^{1/2}$ is the norm of the filtered strain rate tensor, $\bar{\Delta}$ is the filter width, and C_s is the Smagorinsky constant, with values varying from 0.1 to 0.2.¹⁹

In practice, one drawback of the Smagorinsky model is that the constant C_s has to be adjusted to obtain reasonable results. A more universal approach is using the dynamic Smagorinsky model,^{20,21} where the constant C_s is replaced by a local and instantaneous coefficient $C_d(\mathbf{x}, t)$. By using a double filtering process, ie, the original grid filter $\bar{\cdot}$ and a coarser test filter $\tilde{\cdot}$, the coefficient C_d is computed dynamically as

$$C_d = \frac{1}{2} \left(\frac{\mathbf{L} : \mathbf{M}}{\mathbf{M} : \mathbf{M}} \right) = \frac{1}{2} \left(\frac{L_{ij}M_{ij}}{M_{ij}M_{ij}} \right), \quad (6)$$

with

$$\mathbf{L} = \widetilde{\overline{\mathbf{u}\mathbf{u}}} - \overline{\widetilde{\mathbf{u}\mathbf{u}}}, \quad (7)$$

and

$$\mathbf{M} = \bar{\Delta}^{-2} \widetilde{|\bar{\mathbf{S}}|} - \widetilde{\bar{\Delta}^{-2} \widetilde{|\bar{\mathbf{S}}|}} \quad (8)$$

By substituting (4) into (1), the LES governing equations can be rewritten as

$$\begin{cases} \frac{\partial \bar{\mathbf{u}}}{\partial t} + \bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}} = -\nabla \bar{P} + \nabla \cdot (2\nu \bar{\mathbf{S}}) + \nabla \cdot (2\nu_t \bar{\mathbf{S}}), \\ \nabla \cdot \bar{\mathbf{u}} = 0, \end{cases} \quad (9)$$

where \bar{P} is the modified filtered pressure defined as

$$\bar{P} \equiv \frac{\bar{p}}{\rho} + \frac{1}{3} \text{tr} \tau. \quad (10)$$

Dirichlet and Neumann boundary conditions for (9) are

$$\bar{\mathbf{u}} = \mathbf{g} \quad \text{on } \Gamma_D, \quad (11)$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{h} \quad \text{on } \Gamma_N, \quad (12)$$

where Γ_D and Γ_N are complementary subsets of the domain boundary $\Gamma = \partial\Omega$, functions \mathbf{g} and \mathbf{h} are given, \mathbf{n} is the unit outward normal vector of Γ_N , and $\boldsymbol{\sigma} = -\bar{P}\mathbf{I} + 2(\nu + \nu_t)\bar{\mathbf{S}}$ is the Cauchy stress tensor.

As the initial condition, a divergence-free velocity field $\mathbf{u}_0(\mathbf{x})$ is specified over the domain at $t = 0$

$$\bar{\mathbf{u}}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}) \quad \text{in } \Omega. \quad (13)$$

2.2 | Fully implicit finite element discretization

Next, we describe a stabilized finite element discretization of the weak form of the LES equations (9). First, we define the trial function space for the velocity and the scalar function space for the pressure as

$$\begin{aligned} \mathcal{V} &= \{ \mathbf{u}(\mathbf{x}, t) \mid \mathbf{u}(\mathbf{x}, t) \in [H^1(\Omega)]^3, \mathbf{u} = \mathbf{g} \text{ on } \Gamma_D \}, \\ \mathcal{P} &= \{ p(\mathbf{x}, t) \mid p(\mathbf{x}, t) \in L^2(\Omega) \}, \end{aligned} \quad (14)$$

and the weighting function space for the velocity as

$$\mathcal{V}_0 = \{ \mathbf{u}(\mathbf{x}, t) \mid \mathbf{u}(\mathbf{x}, t) \in [H^1(\Omega)]^3, \mathbf{u} = \mathbf{0} \text{ on } \Gamma_D \}, \quad (15)$$

where $H^1(\Omega)$ is the usual Sobolev space. Then, the Galerkin weak form of the LES equations reads as follows: Find $\bar{\mathbf{u}} \in \mathcal{V}$ and $\bar{P} \in \mathcal{P}$ such that

$$B_G(\bar{\mathbf{u}}, \bar{P}; \mathbf{w}, q) = 0 \quad \forall (\mathbf{w}, q) \in \mathcal{V}_0 \times \mathcal{P} \quad (16)$$

with

$$\begin{aligned} B_G(\bar{\mathbf{u}}, \bar{P}; \mathbf{w}, q) &= \left(\frac{\partial \bar{\mathbf{u}}}{\partial t}, \mathbf{w} \right)_\Omega + (\bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}}, \mathbf{w})_\Omega - (\bar{P}, \nabla \cdot \mathbf{w})_\Omega \\ &\quad + (2(\nu + \nu_t)\bar{\mathbf{S}}, \nabla \mathbf{w})_\Omega - (\mathbf{h} \cdot \mathbf{w})_{\Gamma_N} + (\nabla \cdot \bar{\mathbf{u}}, q)_\Omega. \end{aligned} \quad (17)$$

Here, $(\mathbf{f}, \mathbf{g})_\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{g} \, d\Omega$ is the standard scalar inner product in $L^2(\Omega)$.

We discretize the weak form of the LES equations (16) in space with a $P_1 - P_1$ stabilized finite element method.^{22,23} First, we triangulate the computational domain Ω by a conformal tetrahedral mesh $\mathcal{T}_h = \{K\}$ with h_K the diameter of the element $K \in \mathcal{T}_h$. With this, the above-defined spaces (14) and (15) are approximated by finite-dimensional spaces spanned by continuous piecewise linear functions as follows:

$$\begin{aligned} \mathcal{V}^h &= \{ \mathbf{u}^h \mid \mathbf{u}^h \in [C^0(\Omega) \cap H^1(\Omega)]^3, \mathbf{u}^h|_{K \in \mathcal{T}_h} \in P_1(K)^3, \mathbf{u}^h = \mathbf{g} \text{ on } \Gamma_D \}, \\ \mathcal{P}^h &= \{ p^h \mid p^h \in C^0(\Omega) \cap L^2(\Omega), p^h|_{K \in \mathcal{T}_h} \in P_1(K) \}, \end{aligned} \quad (18)$$

and

$$\mathcal{V}_0^h = \{ \mathbf{u}^h \mid \mathbf{u}^h \in [C^0(\Omega) \cap H^1(\Omega)]^3, \mathbf{u}^h|_{K \in \mathcal{T}_h} \in P_1(K)^3, \mathbf{u}^h = \mathbf{0} \text{ on } \Gamma_D \}, \quad (19)$$

where $C^0(\Omega)$ is the set of all continuous functions defined on Ω , and $P_1(K)$ is the space of piecewise linear functions. Because the $P_1 - P_1$ pair does not satisfy the Ladyzenskaja-Babuska-Brezzi inf-sup condition, additional stabilization terms are needed in the formulation. We employ the stabilization technique introduced in the works of Franca and Frey²²

and Whiting and Jansen.²³ The semi-discrete stabilized finite element formulation of (16) reads as follows: Find $\bar{\mathbf{u}}^h \in \mathcal{V}^h$ and $\bar{P}^h \in \mathcal{P}^h$, such that

$$B(\bar{\mathbf{u}}^h, \bar{P}^h; \mathbf{w}^h, q^h) = 0 \quad \forall (\mathbf{w}^h, q^h) \in \mathcal{V}_0^h \times \mathcal{P}^h \quad (20)$$

with

$$\begin{aligned} B(\bar{\mathbf{u}}^h, \bar{P}^h; \mathbf{w}^h, q^h) = & B_G(\bar{\mathbf{u}}^h, \bar{P}^h; \mathbf{w}^h, q^h) + \sum_{K \in \mathcal{T}_h} (\nabla \cdot \bar{\mathbf{u}}^h, \tau_c \nabla \cdot \mathbf{w}^h)_K \\ & + \sum_{K \in \mathcal{T}_h} \left(\frac{\partial \bar{\mathbf{u}}^h}{\partial t} + \bar{\mathbf{u}}^h \cdot \nabla \bar{\mathbf{u}}^h + \nabla \bar{P}^h - 2(\nu + \nu_t) \nabla \cdot \bar{\mathbf{S}}, \tau_m (\bar{\mathbf{u}}^h \cdot \nabla \mathbf{w}^h + \nabla q^h) \right)_K, \end{aligned} \quad (21)$$

where the stabilization parameters τ_c and τ_m are defined as follows:

$$\begin{aligned} \tau_m &= \frac{1}{\sqrt{(2c_1/\Delta t)^2 + (\bar{\mathbf{u}} \cdot G \cdot \bar{\mathbf{u}}) + c_2(\nu + \nu_t)^2 G : G}} \\ \tau_c &= \frac{1}{8\tau_m \text{tr}(G)}. \end{aligned} \quad (22)$$

Here, $G_{ij} = \sum_{k=1}^3 \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j}$ is the covariant metric tensor where $\frac{\partial \xi}{\partial x}$ represents the inverse Jacobian of the mapping between the reference and the physical domain, Δt is the step size for the temporal discretization, and the coefficients c_1 and c_2 are set as: $c_1 = 1$ and $c_2 = 36$.

Integrating the spatial integrals of (20) by the Gauss quadrature, we obtain a time-dependent nonlinear system

$$\frac{dx}{dt} = \mathcal{L}(x). \quad (23)$$

Here, x is the vector of all finite element interpolation coefficients and $\mathcal{L}(x)$ is the nonlinear function of x including all the spatial discretization terms. (23) is further discretized fully implicitly in time with a second-order backward differentiation formula (BDF2):

$$x^n - \frac{4}{3}x^{n-1} + \frac{1}{3}x^{n-2} = \frac{2\Delta t}{3} \mathcal{L}(x^n), \quad (24)$$

where x^n is the value of x at the n th time step, and Δt is the time step size. Only at the first time step, a first-order backward Euler (BDF1) method is used:

$$x^n - x^{n-1} = \Delta t \mathcal{L}(x^n). \quad (25)$$

For simplicity, (24) can be rewritten as a sparse, nonlinear, algebraic system

$$\mathcal{F}^n(x^n) = 0, \quad (26)$$

which has to be solved at each time step to obtain the solution at the next time step. The details of the numerical method for solving this large nonlinear system of equations will be discussed in the next section. We remark here that, there are usually two ways to organized the solution vector, one is using the componentwise (field by field) order where the solution vector x is defined as

$$x = (\bar{u}_0, \bar{u}_1, \dots, \bar{u}_{M-1}, \bar{v}_0, \bar{v}_1, \dots, \bar{v}_{M-1}, \bar{w}_0, \bar{w}_1, \dots, \bar{w}_{M-1}, \bar{P}_0, \bar{P}_1, \dots, \bar{P}_{M-1})^T, \quad (27)$$

and the other one is using the pointwise (field-coupling) order where x is defined as

$$x = (\bar{u}_0, \bar{v}_0, \bar{w}_0, \bar{P}_0, \bar{u}_1, \bar{v}_1, \bar{w}_1, \bar{P}_1, \dots, \bar{u}_{M-1}, \bar{v}_{M-1}, \bar{w}_{M-1}, \bar{P}_{M-1})^T. \quad (28)$$

Here, \bar{u} , \bar{v} , and \bar{w} are x , y , and z components of the velocity $\bar{\mathbf{u}}$, respectively, and M is the total number of mesh points. In our implementation, we adopt the pointwise order instead of the componentwise order which is used by most approaches, the aim is to avoid the large saddle point problem arising from the componentwise ordering when forming the Jacobian system, which will significantly impact the convergence and parallel performance of the algorithm.²⁴ Instead, the pointwise ordering provides a natural setting for the point-block version of incomplete LU (ILU) factorization that is more stable than the classical pointwise ILU and also improves the cache performance of the algorithm.

2.3 | Newton-Krylov-Schwarz algorithm

In this work, we employ an NKS method to solve (26). The algorithm consists of three components, an inexact Newton method together with a cubic line search technique^{25,26} to solve the nonlinear equation, a preconditioned Krylov subspace

method (GMRES)²⁷ to obtain the Newton correction, and a restricted additive Schwarz (RAS) domain decomposition method as a preconditioner²⁸ to accelerate the convergence of the Krylov subspace method. The algorithm goes as follows.

1. Take the solution of the previous time step as the initial guess

$$x_0^n = x^{n-1}. \quad (29)$$

2. For $k = 0, 1, \dots$, until the stopping condition is satisfied:

- (a) Construct the complete Jacobian matrix J_k^n of $\mathcal{F}^n(x)$ at x_k^n .
- (b) Obtain the Newton correction s_k^n by solving the following right-preconditioned Jacobian system inexactly with GMRES:

$$J_k^n (M_k^n)^{-1} M_k^n s_k^n = -\mathcal{F}^n(x_k^n). \quad (30)$$

- (c) Find a step length λ_k^n using a cubic line search and update the approximation as follows:

$$x_{k+1}^n = x_k^n + \lambda_k s_k^n. \quad (31)$$

In our implementation, the nonlinear iteration is stopped if the following condition is satisfied:

$$\|J_k^n s_k^n + \mathcal{F}^n(x_k^n)\| \leq \eta_k \|\mathcal{F}^n(x_k^n)\|, \quad (32)$$

where η_k is the relative tolerance for the linear solver. If η_k is small enough, the algorithm reduces to the exact Newton algorithm. Here, the Jacobian matrix is computed analytically since the robustness of the Newton method is often not guaranteed when the Jacobian is approximately computed. In each individual element, the Jacobian matrix takes the form

$$\begin{pmatrix} K & C \\ G & Q \end{pmatrix}, \quad (33)$$

where K and Q are for the velocity and pressure, respectively, and C and G are the coupling matrix between them. Since there are 12 degrees of freedom (DOFs) for the velocity and 4 DOFs for the pressure in an element, the matrix orders of K , Q , C , and G are 12×12 , 4×4 , 12×4 , and 4×12 , respectively. K , Q , C , and G can be derived by computing the first-order Fréchet derivative of the left-hand side of (20). Let ϕ_i ($i = 1, 2, 3, 4$) denote the piecewise linear basis function for the tetrahedral element, then the weighting functions for the velocity field and the pressure fields will be $\mathbf{w}_i = (\phi_i, 0, 0)$, $\mathbf{w}_{4+i} = (0, \phi_i, 0)$, $\mathbf{w}_{8+i} = (0, 0, \phi_i)$, $i = 1, 2, 3, 4$, and $q_i = \phi_i$, $i = 1, 2, 3, 4$, respectively. Then, the formulations of K , Q , C , and G can be computed as follows:

$$\begin{aligned} K_{ij} = & \alpha(\mathbf{w}_j, \mathbf{w}_i) + (\bar{\mathbf{u}} \cdot \nabla \mathbf{w}_j + \mathbf{w}_j \cdot \nabla \bar{\mathbf{u}}, \mathbf{w}_i) + ((\nu + \nu_t)(\nabla \mathbf{w}_j + \nabla \mathbf{w}_j^T), \nabla \mathbf{w}_i) + (\nabla \cdot \mathbf{w}_j, \tau_c \nabla \cdot \mathbf{w}_i) \\ & + \alpha(\mathbf{w}_j, \tau_m \bar{\mathbf{u}} \cdot \nabla \mathbf{w}_i) + (\Delta_t \bar{\mathbf{u}}, \tau_m \mathbf{w}_j \cdot \nabla \mathbf{w}_i) + (\bar{\mathbf{u}} \cdot \nabla \mathbf{w}_j + \mathbf{w}_j \cdot \nabla \bar{\mathbf{u}}, \tau_m \bar{\mathbf{u}} \cdot \nabla \mathbf{w}_i) \\ & + (\bar{\mathbf{u}} \cdot \nabla \bar{\mathbf{u}}, \tau_m \mathbf{w}_j \cdot \nabla \mathbf{w}_i) + (\nabla \bar{P}, \tau_m \mathbf{w}_j \cdot \nabla \mathbf{w}_i), \end{aligned} \quad (34)$$

$$Q_{ij} = (\nabla q_j, \tau_m \nabla q_i), \quad (35)$$

$$C_{ij} = -(q_j, \nabla \cdot \mathbf{w}_i) + (\nabla q_j, \tau_m \bar{\mathbf{u}} \cdot \nabla \mathbf{w}_i), \quad (36)$$

and

$$G_{ij} = (\nabla \cdot \mathbf{w}_j, q_i) + \alpha(\mathbf{w}_j, \tau_m \nabla q_i) + (\bar{\mathbf{u}} \cdot \nabla \mathbf{w}_j + \mathbf{w}_j \cdot \nabla \bar{\mathbf{u}}, \tau_m \nabla q_i), \quad (37)$$

where

$$\alpha = \begin{cases} \frac{1}{\Delta t}, & \text{for backward Euler,} \\ \frac{3}{2\Delta t}, & \text{for BDF2,} \end{cases} \quad (38)$$

and

$$\Delta_t \bar{\mathbf{u}} = \begin{cases} \frac{\bar{\mathbf{u}}^n - \bar{\mathbf{u}}^{n-1}}{\Delta t}, & \text{for backward Euler,} \\ \frac{3\bar{\mathbf{u}}^n - 4\bar{\mathbf{u}}^{n-1} + \bar{\mathbf{u}}^{n-2}}{2\Delta t}, & \text{for BDF2.} \end{cases} \quad (39)$$

Here, we have assumed that the unknowns are ordered field by field for the element Jacobian matrix for sake of simplicity in the notation. Remember that the unknowns of the global solution are organized in pointwise order, therefore, it should be careful when assembling the global Jacobian matrix from the element Jacobian matrices.

The preconditioner $(M_k^n)^{-1}$ is critically important, without which the GMRES usually converges slowly or does not converge, and as a result, the outer inexact Newton may not converge well either. An overlapping RAS preconditioner²⁸ is employed in our algorithm. The preconditioning matrix $(M_k^n)^{-1}$ is constructed as follows. First, we partition the computational domain Ω into n_p nonoverlapping subdomains Ω_ℓ ($\ell = 1, \dots, n_p$), where n_p is the same as the number of processors. Then, each subdomain is extended to an overlapping subdomain Ω_ℓ^δ by including δ layers of mesh elements from its adjacent subdomains. Here, δ is an integer indicating the level of overlap.

On each overlapping subdomain Ω_ℓ^δ , we define the restriction operator R_ℓ^δ to be the matrix that maps the global vector of unknowns in Ω to those belonging to Ω_ℓ^δ , such that

$$x_\ell^\delta = R_\ell^\delta x = \begin{pmatrix} I & 0 \\ 0 & x \backslash x_\ell^\delta \end{pmatrix}. \quad (40)$$

Here, $x \backslash x_\ell^\delta$ means the unknowns outside the subdomain Ω_ℓ^δ . For simplicity, the subscript n is dropped here without confusion. We then construct a subdomain Jacobian matrix by

$$J_\ell^\delta = R_\ell^\delta J_k^n (R_\ell^\delta)^T, \quad \ell = 1, 2, \dots, n_p, \quad (41)$$

which is the restriction of the global Jacobian matrix J_k^n to the subdomain Ω_ℓ^δ . Here, $(R_\ell^\delta)^T$ is the extension operator, which is defined as the transpose of the restriction operator R_ℓ^δ . Using these definitions, the RAS preconditioner is defined as

$$(M_k^n)^{-1} = \sum_{\ell=1}^{n_p} (R_\ell^\delta)^T (J_\ell^\delta)^{-1} R_\ell^\delta, \quad (42)$$

where R_ℓ^0 is the restriction operator to the unknowns in the nonoverlapping subdomain Ω_ℓ , defined similarly as R_ℓ^δ , and $(J_\ell^0)^{-1}$ is the inverse of the subdomain Jacobian J_ℓ^0 . In practice, $(J_\ell^0)^{-1}$ is obtained by solving a subdomain linear system. Because $(J_\ell^0)^{-1}$ is used as a preconditioner here, it can be solved exactly or approximately. In our application, it is solved with a point-block incomplete LU factorization with some levels of fill-ins.²⁹

3 | NUMERICAL RESULTS AND PARALLEL PERFORMANCE

In this section, we first provide a validation of the proposed fully implicit discretization scheme using a lid-driven cavity flow problem and a square cylinder flow problem, both have been studied by many researchers, and then, we perform an LES of flows around a high-speed train and investigate the numerical behavior and the parallel performance of the proposed algorithm. Our algorithm is implemented on top of the Portable Extensible Toolkit for Scientific computing (PETSc) library.³⁰ The unstructured tetrahedral mesh is generated by using ANSYS ICEM, and the mesh partition for parallel computing is obtained with the software ParMETIS. All computations are carried out on TianHe-2 supercomputer at National Supercomputer Center in Guangzhou, China. The relative stopping conditions for the nonlinear solver and linear solver are set to be 10^{-6} and 10^{-4} , respectively, for all the test cases. Note that the nonlinear equation is solved by an inexact Newton method, where the linear solver is only used to compute the Newton correction; therefore, the stopping condition for the linear solver can be coarser here.

3.1 | Validation of the proposed numerical method

To validate the proposed fully implicit discretization and solver, we consider two simple test cases that have been well-understood: a lid-driven cavity flow problem and a square cylinder flow problem.

First, we perform an LES for the lid-driven cavity flow. The detailed geometry of the problem is sketched in Figure 1. We consider a fluid with constant viscosity μ and constant density ρ in a square cavity with length L , depth D , and height H . The flow is driven by the wall at $y = H$ that moves tangentially in the x direction with a constant velocity U . The dimensionless Reynolds number is defined as $\text{Re} = \rho UL / \mu$, based on the driven velocity, and the length of the cavity as the characteristics of the velocity and length, respectively.

In this test, we set $L = D = H = 1$ and $\text{Re} = 10^4$. We run our simulation on a mesh with about 8.04×10^5 elements (total degree of freedom $\text{DOF} = 6.81 \times 10^5$) and with $\Delta t = 0.01$, and we use both the Smagorinsky model with $C_s = 0.17$ and the dynamic Smagorinsky model. The level of fill-ins for the linear solver is 1 and the overlapping size for the domain decomposition is 2. The time duration for the simulation is 80, and we computed the mean velocity by time averaging

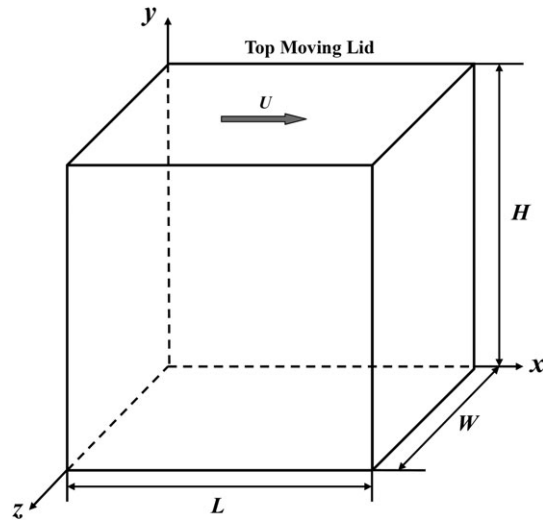


FIGURE 1 Sketch of the geometry of the lid-driven cavity

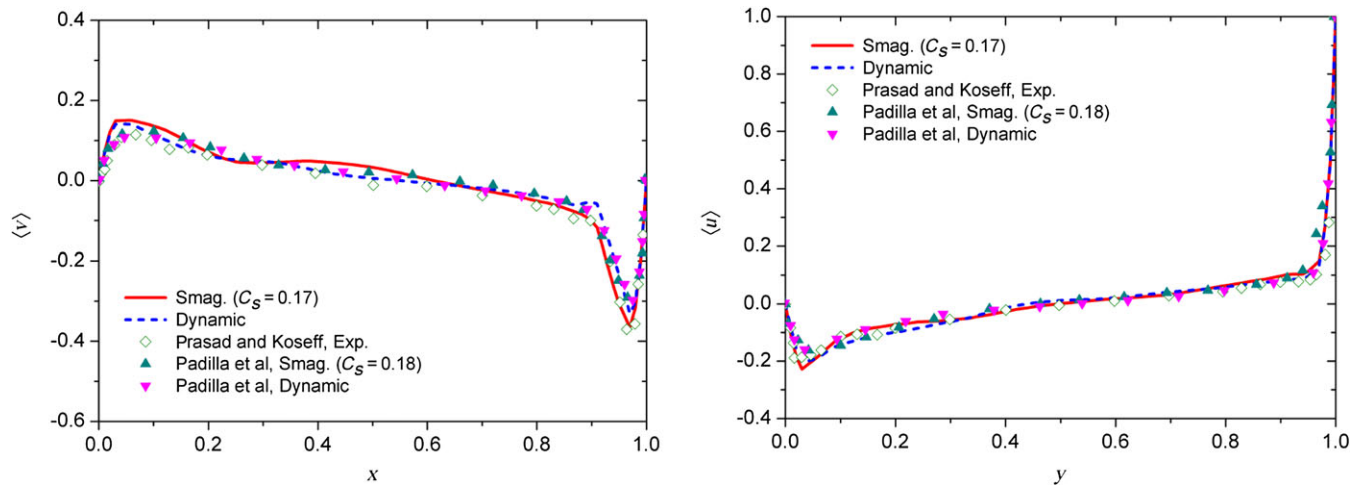


FIGURE 2 Comparison of the mean velocity in the midplane $z = 1/2$. Left: $\langle v \rangle(x, 1/2, 1/2)$; right: $\langle u \rangle(1/2, y, 1/2)$ [Colour figure can be viewed at wileyonlinelibrary.com]

from $t = 40$ to 80 . The numerical results are compared with the experimental data in the work of Prasad and Koseff³¹ and the LES results in the work of Padilla et al.³² Figure 2 shows that the mean velocity profiles of all the results are nearly the same.

Next, we simulate a flow passing a square cylinder at $Re = 2.2 \times 10^4$. The computational domain is shown in Figure 3, where $D = 1$ is the diameter of the square cylinder, $W = 14D$ and $H = 4D$ are the width and height of the domain, respectively, $L_i = 5D$ is the distance between the cylinder and the inflow surface, and $L_o = 21D$ is the distance between the cylinder and the outflow surface. The Reynolds number is defined as $Re = U_\infty D / \nu$. We run the simulation on a mesh with about 1.97×10^6 elements ($DOF = 1.43 \times 10^6$) and with $\Delta t = 0.01$, and use both the Smagorinsky model with $C_s = 0.17$ and the dynamic Smagorinsky model. The level of fill-ins and the overlapping size are also set to 1 and 2, respectively. The time duration for the simulation is 200. The flow is full developed after $t = 100$ (the periods of one vortex shedding is approximately 8.0).

In Table 1, we show the computed values of the mean drag coefficient C_D , the base pressure coefficient C_{PB} , the root mean square value of the lift coefficient $C_{L,rms}$, and the Strouhal numbers St . These values were obtained by time averaging from $t = 100$ to 200 . In the same Table, we also present results obtained by others and by experiments. It is clear that our results agree very well with those published results.

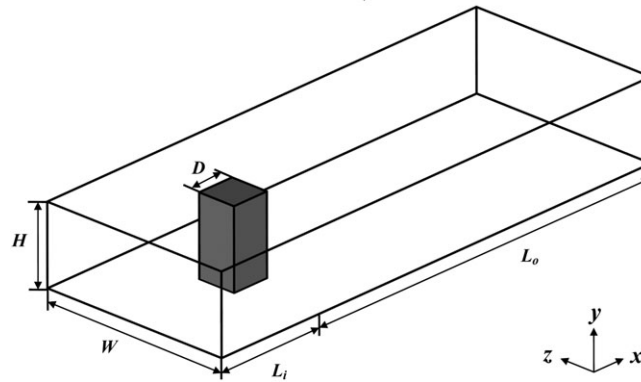


FIGURE 3 Dimensions of the computational domain

TABLE 1 Comparison of the mean drag coefficient C_D , the base pressure coefficient C_{PB} , the root mean square values of the lift coefficient $C_{L,rms}$, and the Strouhal numbers St

	C_D	C_{PB}	$C_{L,rms}$	St
Smagorinsky	2.24	-1.41	1.18	0.132
Dynamic	2.31	-1.52	1.27	0.133
Smagorinsky (Rodi et al ³³)	1.66-2.77	-	0.38-1.79	0.07-0.15
Dynamic LES (Sohankar et al ³⁴)	2.00-2.32	1.30-1.63	1.23-1.54	0.127-0.135
VMS-LES (Koobus and Farhat ³⁵)	2.10	1.52	1.08	0.136
<i>Experimental results</i>				
Lyn et al ³⁶	2.05-2.25	-	-	0.132
Bearman and Obasaju ³⁷	2.28	-1.60	1.20	0.13
Kogaki et al ³⁸	2.22	-1.64	-	0.125

Abbreviations: LES, large eddy simulation; VMS-LES, variational multiscale-large eddy simulation.

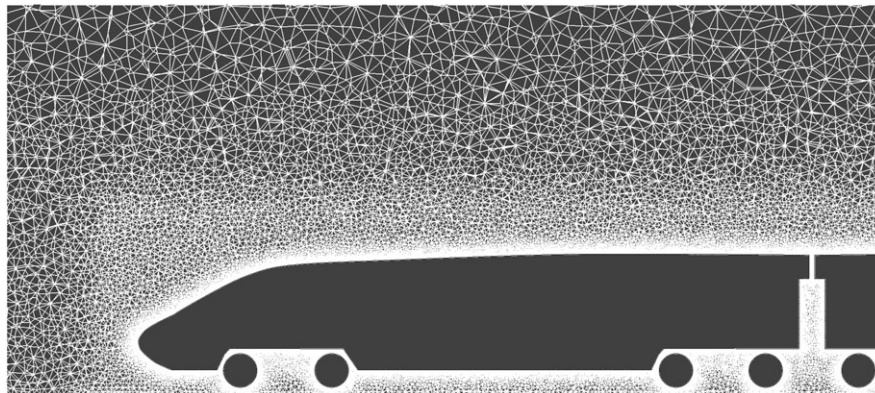


FIGURE 4 Mesh around the head the high-speed train

3.2 | LES of flows around a high-speed train

In this section, we perform an LES for flows around a high-speed train with realistic geometry. The train model is derived from China's high-speed train CRH380B with eight cars, which is designed for passenger transportation with an operational speed of 300 km/h and a maximum speed of 380 km/h. The length, width, and height of the train are 182 m, 3.26 m, and 3.39 m, respectively. The simulation is carried out with a mesh of 1.90×10^7 elements ($DOF = 1.42 \times 10^7$). The mesh size on the train is set to be 0.016, a density box with a mesh size of 0.2 is created around the train, where the width between the train and the density box is set to be 1.0, and the mesh size on other region is set to be 1.6. Grid space on the wall is stretched with an expansion factor of 1.3 in the normal direction. The mesh around the head of the train is illustrated in Figure 4. The train runs at a speed of 100 m/s (360 km/h) and experiences a crosswind with a speed of 17.6 m/s

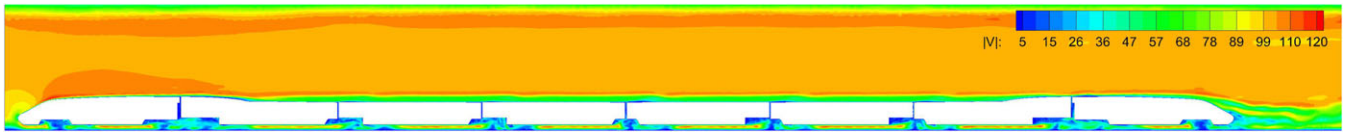


FIGURE 5 Velocity distribution in the intermediate cross-section of the train [Colour figure can be viewed at wileyonlinelibrary.com]

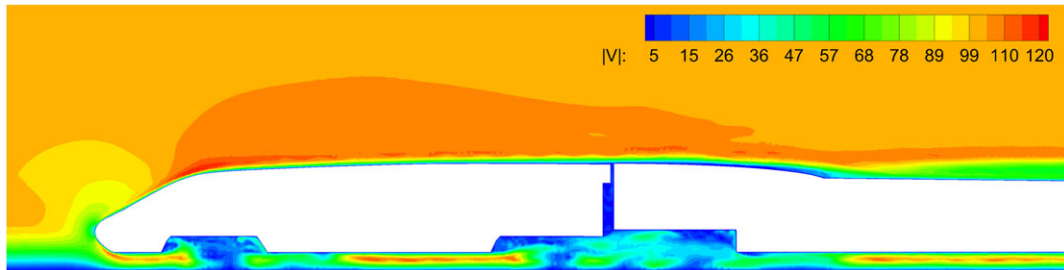


FIGURE 6 Velocity distribution near the head of the train [Colour figure can be viewed at wileyonlinelibrary.com]

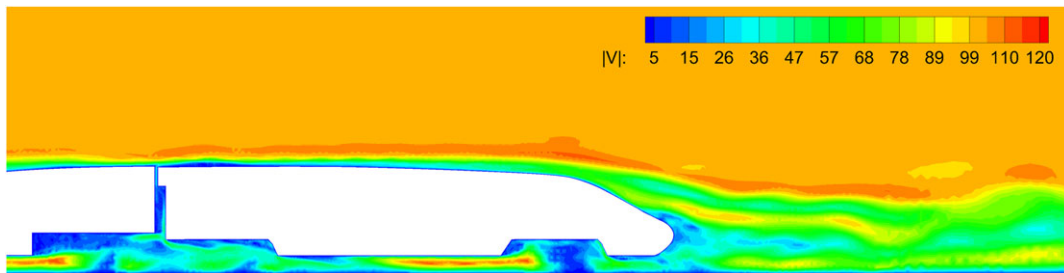


FIGURE 7 Velocity distribution near the tail of the train [Colour figure can be viewed at wileyonlinelibrary.com]

(the yaw angle is 10°). The fluid material we adopt is air at 25°C with density $\rho = 1.185 \text{ kg/m}^3$ and dynamic viscosity $\mu = 1.831 \times 10^{-5} \text{ kg/ms}$. The Reynolds number is $\text{Re} = 2.14 \times 10^7$, which is defined as

$$\text{Re} = \frac{\rho U_\infty W}{\mu},$$

where $U_\infty = 101.537 \text{ m/s}$ is the freestream velocity that is equal to the effective crosswind velocity, the characteristic length is the width of the train $W = 3.26 \text{ m}$.

The numerical simulation of flows around a high-speed train is a very challenging problem because of the very high Reynolds number and the slender body of the train, which leads to a very complex flow field with high turbulent intensities and large separation. In this work, we focus on the robustness and the parallel performance of the proposed solver for LES.

The contours of the wind velocity magnitude in the intermediate cross-section of the train at $t = 1$ second (time step size $\Delta t = 0.004$ seconds) is shown in Figure 5, and more details of the flow field around the head and the tail of the train can be viewed in Figure 6 and Figure 7. It can be seen that the flow separates at the windward corner of the head first and then reattaches and stays attached on the roof and finally detaches close to the train's downwind surface and becomes turbulent wake behind the tail of the train. Figure 8 shows the stream trace of the flow near the tail of the train, and Figure 9 shows the wake vortex structure illustrated by the Q-criteria.

As the train experiences a crosswind, a series of inclined wake vortex shedding from the leeward side of the train can be observed, as shown in the contour of the wind velocity magnitude at different height (Figure 10) and that in the cross-sections at different locations (Figure 11).

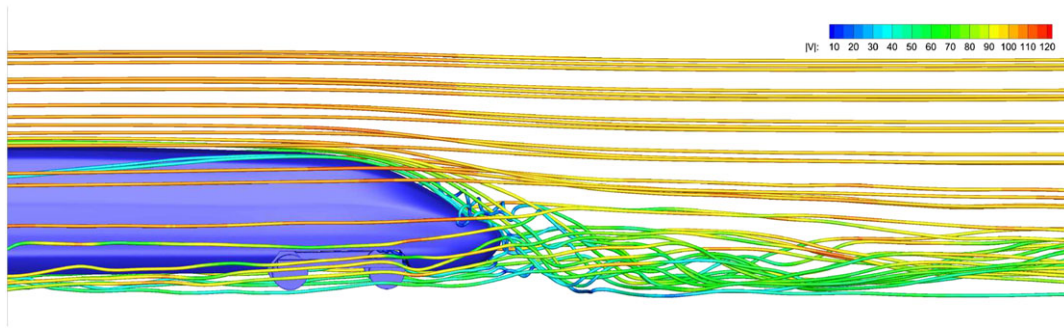


FIGURE 8 The stream trace of the flow near the tail of the train [Colour figure can be viewed at wileyonlinelibrary.com]

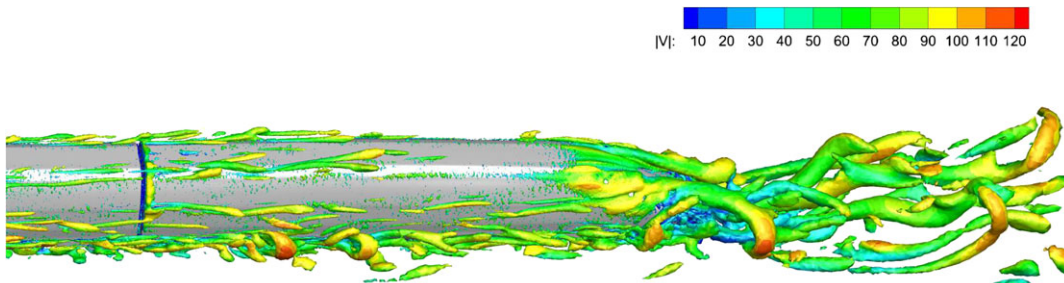


FIGURE 9 Instantaneous vortex structure around the tail of the train illustrated by the Q-criteria, colored by the velocity magnitude ($Q = 1000$) [Colour figure can be viewed at wileyonlinelibrary.com]

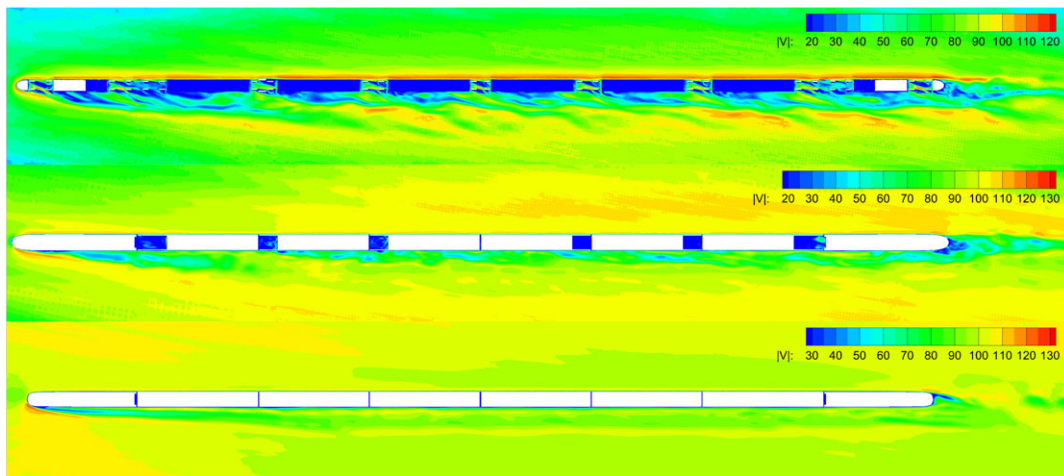


FIGURE 10 Velocity distribution of the cross-section at the height of $z = 0$, $z = 1$ and $z = 3$ (from top to bottom) [Colour figure can be viewed at wileyonlinelibrary.com]

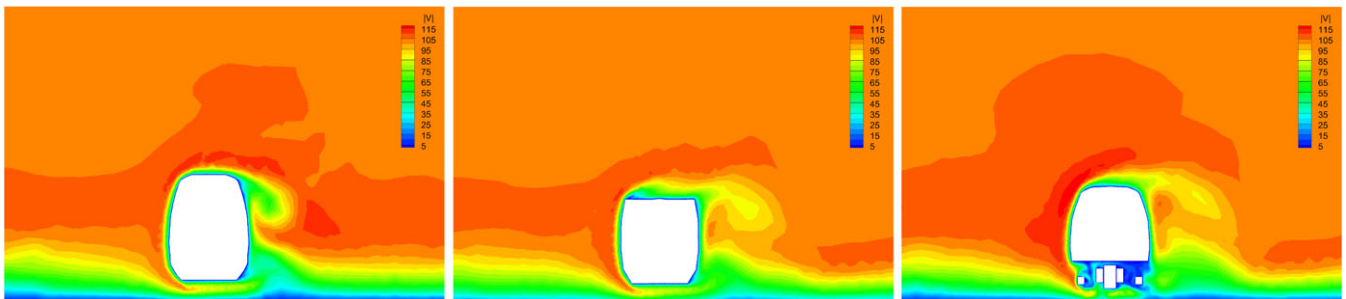


FIGURE 11 Velocity distribution of the cross-section at different locations: $y = 15$, $y = 75$, and $y = 180$ (from left to right) [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Performance results on using different time step sizes for the fully implicit method. The simulation is carried out with a mesh of 1.90×10^7 elements using 512 processors. “Newton” denotes the average Newton iterations per time step, “GMRES” denotes the average GMRES iterations per Newton step, “Time/Step” refers to the average computing time per time step in seconds, and “CFL” denotes the CFL number

Δt	Newton	GMRES	Time/Step	CFL
5.0×10^{-4}	2.0	42.66	23.64	1.41
1.0×10^{-3}	3.0	68.82	44.20	2.81
2.0×10^{-3}	3.0	76.96	47.78	5.56
4.0×10^{-3}	3.0	84.70	51.27	11.0
8.0×10^{-3}	4.0	96.01	74.26	21.8
1.6×10^{-2}	4.4	133.55	107.58	43.6
3.2×10^{-2}	6.2	152.51	170.08	86.8

Abbreviation: CFL, Courant-Friedrichs-Lewy.

3.3 | Performance of the fully implicit method

In this section, we report the numerical behavior and parallel performance of the proposed fully implicit method. We first make a stability test of the fully implicit method in terms of the CFL numbers and then investigate several issues including the strong scalability study, the influence of different overlaps and subdomain solvers, and the influence of the accuracy of the Jacobian. The case of the high-speed train is used for all the following tests.

3.3.1 | Stability test of the fully implicit method

The CFL stability condition is not required by the fully implicit method. Therefore, compared with explicit and semi-implicit methods, the fully implicit method is usually more stable with a much larger time step size, which depends only on the accuracy requirement. We show the stability of the proposed implicit scheme numerically by investigating its performance on using different time steps. The results on the average Newton iterations per time step, the average GMRES iterations per Newton step, the average computing time per time step, as well as the CFL numbers are shown in Table 2. Here, the documented results are averaged values over 20 time steps after the flow fully developed, and the CFL number is calculated via

$$\text{CFL} = \max \left\{ \frac{|V_K| \Delta t}{h_K} \right\},$$

where h_K is the cell size and $|V_K|$ is the magnitude of the velocity through that cell.

Table 2 shows that both the number of Newton iterations and the number of GMRES iteration increase gradually as the time step size is increased, and in result, the computing time per time step also increases. However, since the increment rate of the computing time per time step is much slower than the time step size, we can infer that a larger time step will save the total computing time in overall. The allowed largest time step size is $\Delta t = 0.032$, which corresponding to a CFL number of 86.8. Therefore, we see that, by using the fully implicit method, the time step size is no longer constrained by the CFL condition.

For the purpose of comparison, we also implemented a second-order semi-implicit BDF method introduced by Forti and Dedè,⁸ where the nonlinear terms of the LES equations are linearized by extrapolating the solution from the previous time steps based on Newton-Gregory backward polynomials. The velocity variables at the n th time step are expressed by

$$\bar{\mathbf{u}}^n = 2\bar{\mathbf{u}}^{n-1} - \bar{\mathbf{u}}^{n-2}, \text{ for BDF2.} \quad (43)$$

With this extrapolations, the discretization system of the LES equations is linearized and we do not need to solve the nonlinear equations. However, it should be note that we still have to solve a linear algebraic equation at each time step due to the incompressibility constraint. We employ a GMRES method to solve this linear system, the stopping condition is set to be 10^{-6} , which is the same with the implicit solver. The performance result is shown in Table 3. It is observed that this semi-implicit method is more stable than explicit schemes, since a maximum CFL number of 1.41 is obtained,

TABLE 3 Performance results on using different time step sizes for the semi-implicit method. The notations are the same with Table 2

Δt	GMRES	Time/Step	CFL
1.25×10^{-4}	31.6	6.14	0.39
2.5×10^{-4}	41.0	6.75	0.75
5.0×10^{-4}	64.6	8.34	1.41

Abbreviation: CFL, Courant-Friedrichs-Lewy.

which is generally less than 1 for explicit schemes. When using the same time step size $\Delta t = 5.0 \times 10^{-4}$, the computing time of the semi-implicit method is nearly 1/3 of that of the fully implicit method. Therefore, a fully implicit scheme is usually more expensive than the explicit and semi-implicit methods due to the need for solving nonlinear systems at each time step. However, since much larger time step size is allowed for the fully implicit scheme, the total computing time for the same duration of calculation can be shorter. For example, if the accuracy is not take into account, the total computing time of the fully implicit method is about 1/3 of that of the semi-implicit scheme, when both using the allowed largest time step size. Overall, one of the advantages of the fully implicit approach is that it can solve exactly the nonlinear problem without linearized approximation, and the other one is the possibility of using a large time step, which is useful for long-time simulations when the short time scales of the flows are not important.

3.3.2 | Strong scalability

Next, we study the parallel performance and scalability of the proposed algorithm. To investigate the impact of the amount of the mesh on the parallel performance, two different meshes, one with about 9.61×10^6 elements ($\text{DOF} = 7.17 \times 10^6$) and the other with about 1.90×10^7 elements ($\text{DOF} = 1.42 \times 10^7$) are employed for the test. The fine mesh is the main mesh that we used for most of our simulations, and the coarse mesh with about half of DOFs is used for comparison in testing the parallel scalability. A fixed time step $\Delta t = 0.004$ is applied for all the tests. We report the average computing times and the number of nonlinear iterations per time step, as well as the average GMRES iterations per Newton step in the tests, where the documented results are averaged values over 20 time steps.

Table 4 and Figure 12 show the parallel performance of the algorithm. As the number of processors increases, the number of Newton iterations per time step does not change, the number of GMRES iterations increases slightly, and the computing time decreases quickly. In Figure 12, the parallel speedup is shown to be nearly linear with up to 4096 processors. The parallel efficiency is 63% for the case of $\text{DOF} = 7.17 \times 10^6$ and 71% for the case of $\text{DOF} = 1.42 \times 10^7$ when the number of processors increase from 256 to 4096. Note that, when the same number of processors is used, the

TABLE 4 Parallel performance of the NKS algorithm. Here, the overlapping size for RAS is $\delta = 2$, and the level of ILU fill-ins is $l = 1$

n_p	Newton	GMRES	Time	Speedup	Ideal	Efficiency
Mesh 1: DOF = 7.17×10^6						
256	3.0	73.7	45.56	1	1	100%
512	3.0	74.1	23.99	1.90	2	95%
1024	3.0	75.5	12.91	3.53	4	88%
2048	3.0	77.3	7.46	6.11	8	76%
4096	3.0	80.2	4.54	10.04	16	63%
Mesh 2: DOF = 1.42×10^7						
256	3.0	83.4	98.43	1	1	100%
512	3.0	84.7	51.27	1.92	2	96%
1024	3.0	85.7	27.36	3.60	4	90%
2048	3.0	87.7	14.81	6.65	8	83%
4096	3.0	89.5	8.65	11.38	16	71%

Abbreviations: NKS, Newton-Krylov-Schwarz; RAS, restricted additive Schwarz.

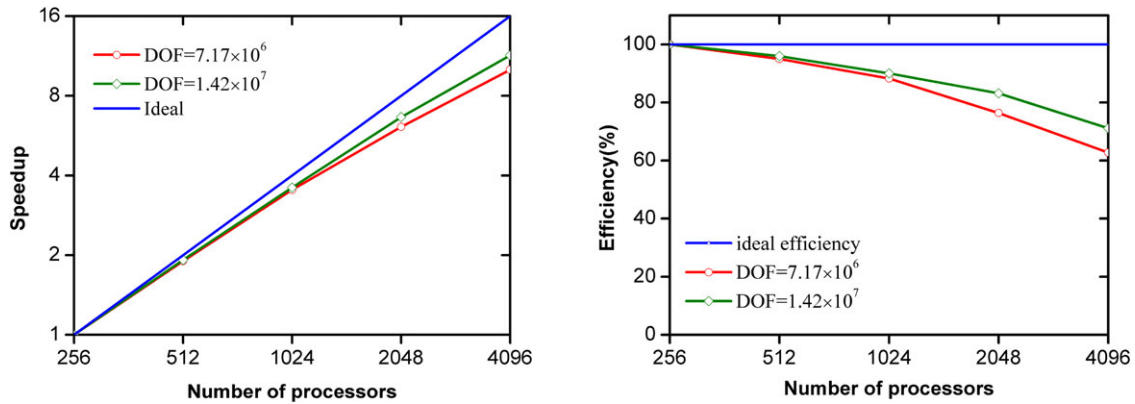


FIGURE 12 The parallel speedup and the parallel efficiency for the high-speed train simulation [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 5 Impact of the local linear solver on the NKS algorithm. Here, the total number of unknowns is $\text{DOF} = 7.17 \times 10^6$, the time step size is $\Delta t = 0.004$, and the overlapping size for RAS is $\delta = 2$

n_p	Newton			GMRES			Time		
	ILU(0)	ILU(1)	ILU(2)	ILU(0)	ILU(1)	ILU(2)	ILU(0)	ILU(1)	ILU(2)
256	3.70	3.00	3.00	97.34	73.73	58.63	54.07	45.56	50.95
512	3.60	3.00	3.00	94.41	74.13	60.17	28.01	23.99	27.48
1024	3.50	3.00	3.00	100.3	75.45	62.15	14.77	12.91	14.81
2048	3.60	3.00	3.00	97.10	77.36	65.42	8.65	7.46	8.23

Abbreviations: NKS, Newton-Krylov-Schwarz; RAS, restricted additive Schwarz.

size of the subdomain of Mesh 2 is about twice of Mesh 1, which means that the cost of communication relative to the computation is smaller for Mesh 2, therefore, the speedup performance of Mesh 2 is a little better.

3.3.3 | Influence of subdomain solvers and different overlaps

In an overlapping Schwarz preconditioner, the choice of subdomain solver has a remarkable impact on the overall performance. Table 5 shows the numerical performance with respect to different subdomain solvers that use different level of fill-in l . It is observed that the number of Newton iterations does not change much with respect to the fill-in level, and as expected, the number of GMRES iterations decreases dramatically as the level of fill-in increases, since a higher level of fill-ins leads to a stronger preconditioner. However, at the same time, more computing time is spent on the ILU factorization in the preconditioning stage. In this test, the ideal level of fill-in is $l = 1$, with which the computing time per time step reaches the minimum.

For the additive Schwarz preconditioner, an important parameter that influences the effects of the preconditioner is the overlapping size δ . In Table 6, we show the results with different choices of δ . Overall, a larger δ implies a faster convergence of GMRES. However, the size of the subdomain problem also grows as δ increases, which implies that more communication time and more computing time are spent on the preconditioning stage. Therefore, there is a trade-off between the number of GMRES iterations and the communication and computing time. The best result is obtained with $\delta = 2$.

3.3.4 | Influence of the accuracy of the Jacobian

In our implementation, we compute the Jacobian matrix analytically including all the terms of \mathcal{F} defined by the left-hand side of (20) including the terms of LES equations and the stabilization terms, and as a result, the inexact Newton converges well with only a small number of iterations. An interesting question is that how Newton converges when the Jacobian is not accurately computed, for example, if we do not include all the terms of \mathcal{F} when computing the Jacobian. For this

TABLE 6 Impact of the overlapping parameter δ on the NKS algorithm. Here, the total number of unknowns is $\text{DOF} = 7.17 \times 10^6$, the time step size is $\Delta t = 0.004$, and the subdomain solver is ILU(1). The simulation is carried out with 1024 processors

δ	Newton	GMRES	Time
0	3.00	96.20	13.06
1	3.00	80.96	13.05
2	3.00	75.45	12.91
3	3.10	74.69	18.84
4	3.00	71.37	18.03

Abbreviation: NKS, Newton-Krylov-Schwarz.

TABLE 7 The influence of the Jacobian on the convergence of the NKS algorithm. The analytical Jacobian means that the Jacobian is analytically computed, whereas the approximated Jacobian means that the term of LES model is not included in the Jacobian

	$\Delta t = 0.004$			$\Delta t = 0.04$		
	Newton	GMRES	Time	Newton	GMRES	Time
Analytical Jacobian	3.0	74.13	23.99	4.7	101.0	40.78
Approximated Jacobian	4.1	67.71	29.74	21.3	77.72	176.8

Abbreviations: LES, large eddy simulation; NKS, Newton-Krylov-Schwarz.

TABLE 8 Performance of the NKS algorithm when the Jacobian matrix is reused for several iterations. Here, the Jacobian lag denotes the number when the Jacobian is rebuilt in the nonlinear solve. ∞ means that the Jacobian is built only once for every time step

Jacobian Lag	$\Delta t = 0.004$			$\Delta t = 0.04$		
	Newton	GMRES	Time	Newton	GMRES	Time
1	3.0	74.13	23.99	4.7	101.0	40.78
2	3.0	67.97	20.28	9.0	76.10	57.60
3	3.1	68.87	17.68	11.3	69.01	63.92
4	3.1	68.87	17.23	14.9	66.54	80.87
∞	3.1	68.87	17.29	DIVERGENCE		

Abbreviation: NKS, Newton-Krylov-Schwarz.

purpose, we conduct a numerical experiment on the convergence of the algorithm when the terms of subgrid model are not included in the Jacobian. The results together with those with the analytical Jacobian are shown in Table 7.

Two cases with different time step size are tested. For the case of $\Delta t = 0.004$, the number of Newton iterations and the total computing time both have small increase when the approximated Jacobian is adopted. However, for $\Delta t = 0.04$, the number of Newton iterations and the total computing time both increase a lot by more than 4 times from those of analytical Jacobian. When we continue to increase the time step size to $\Delta t = 0.08$, the Newton method fails to converge. These results show that using an approximated Jacobian may deteriorate the overall convergence, depending on the time step size, or in other words, the degree of the change of the solution.

In our implementation, the Jacobian matrix is recomputed for every Newton iteration and the preconditioner is recomputed for every Jacobian solve. Next, we investigate the performance of the algorithm when the Jacobian matrix and/or the preconditioner are reused for several iterations. The results are shown in Tables 8 to 10.

TABLE 9 Performance of the NKS algorithm when the preconditioner is reused for several iterations. Here, the preconditioner lag denotes the number when the preconditioner is rebuilt in the nonlinear solve. ∞ means that the preconditioner is built only once for every time step

Preconditioner Lag	$\Delta t = 0.004$			$\Delta t = 0.04$		
	Newton	GMRES	Time	Newton	GMRES	Time
1	3.0	74.13	23.99	4.7	101.0	40.78
2	3.0	74.13	23.38	4.7	124.41	41.20
3	3.0	74.17	23.17	4.7	146.49	45.23
4	3.0	74.17	23.26	4.7	167.39	48.88
∞	3.0	74.17	23.30	4.7	175.44	50.02

Abbreviation: NKS, Newton-Krylov-Schwarz.

TABLE 10 Performance of the NKS algorithm when both the Jacobian matrix and the preconditioner are reused for several iterations. Here, the lag denotes the number when the Jacobian and preconditioner are rebuilt in the nonlinear solve. ∞ means that both the Jacobian and the preconditioner are built only once for every time step

Lag	$\Delta t = 0.004$			$\Delta t = 0.04$		
	Newton	GMRES	Time	Newton	GMRES	Time
1	3.0	74.13	23.99	4.7	101.0	40.78
2	3.0	67.97	19.99	9.0	76.10	57.34
3	3.1	68.87	17.77	11.3	69.01	64.64
4	3.1	68.87	17.45	14.9	67.66	79.09
∞	3.1	68.87	17.42	DIVERGENCE		

Abbreviation: NKS, Newton-Krylov-Schwarz.

Table 8 shows that, when the Jacobian is rebuilt for several Newton iterations, the number of Newton iterations does not change much and the total computing time decreases a little for $\Delta t = 0.004$, while both of them increase dramatically for $\Delta t = 0.04$. What is more, the algorithm fails to converge when the Jacobian is built only once for every time step, and used for all Newton iterations within a time step. These results are in line with those when the Jacobian matrix is approximately computed as shown in Table 7.

Table 9 shows that, when the preconditioner is rebuilt for several Newton iterations while the Jacobian is rebuilt for every Newton iteration, the number of Newton iterations, the number of GMRES iterations, and the total computing time are all the same for the case of $\Delta t = 0.004$. But for the case of $\Delta t = 0.04$, while the number of Newton iterations does not change, the number of GMRES iterations and the total computing time increases notably. It implies that, when the solution is changing rapidly, the accuracy of the preconditioner may dramatically affect the convergence rate of GMRES.

When both the Jacobian matrix and the preconditioner are reused for several iterations, the results are similar to those when only the Jacobian matrix are reused for several iterations, as shown in Tables 8 and 10.

To further understand these results, we investigate the computing time spent on the evaluation of the function, the evaluation of the Jacobian matrix, the construction of the preconditioner, and the linear solver, as shown in Figure 13. We see that the computing time on the construction of the preconditioner is only a very small portion of the total computing time, therefore, rebuilding the preconditioner for several Newton iterations does not save much time. On the other hand, the time spent on the evaluation of the Jacobian is considerable, thus it has a great impact on the performance of the algorithm. For the case of $\Delta t = 0.004$, rebuilding the Jacobian for every 3 Newton iterations reduces a lot of time on the evaluation of the Jacobian, and thus reduces the total computing time as well. However, for the case of $\Delta t = 0.04$, the time spent on the evaluation of the Jacobian increases when the Jacobian is rebuilt for every 3 Newton iterations, the reason is that the number of Newton iterations increases a lot at the same time.

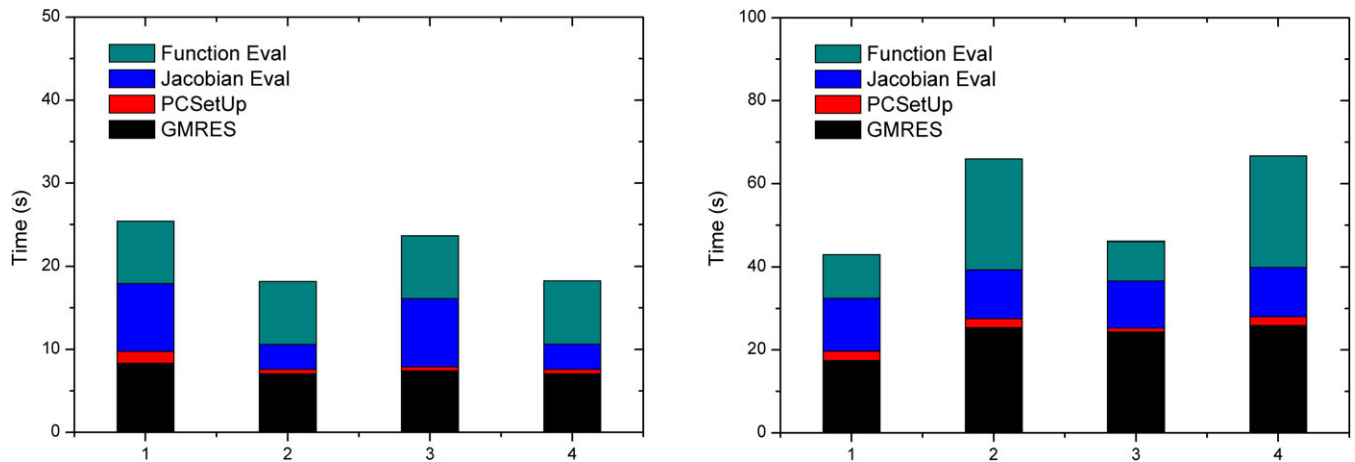


FIGURE 13 The computing times spent on the evaluation of the function, the evaluation of the Jacobian matrix, the construction of the preconditioner, and the linear solver. Columns 1 to 4 indicate the cases of a standard solver, Jacobian is rebuilt for every 3 iterations, preconditioner is rebuilt for every 3 iterations, and both the Jacobian and the preconditioner are rebuilt for every 3 iterations, respectively. Left: the case of $\Delta t = 0.004$; right: the case of $\Delta t = 0.04$ [Colour figure can be viewed at wileyonlinelibrary.com]

In summary, we remark that, in the proposed algorithm, an exact Jacobian can improve the convergence of the Newton method and make the algorithm more robust. Although in some cases, we can reduce the overall computing time by using an approximated Jacobian or reusing the Jacobian for several Newton iterations, but such treatments do not work and the algorithm may fail to converge when the solution changes sharply (eg, for a large time step). Therefore, it is worthwhile to compute the Jacobian analytically even though it is rather a sophisticated task, since it will save many Newton iterations, and can be used to provide a better preconditioner for the Jacobian systems.

4 | CONCLUSION

In this paper, we developed a fully implicit finite element method on unstructured meshes for the LES of incompressible turbulent flows and a scalable parallel domain decomposition method for solving the large nonlinear algebraic system of equations arising from the discretization. The Smagorinsky and dynamic Smagorinsky models are adopted for the modeling of small eddies in turbulent flows. We validated the proposed numerical method by solving two benchmark problems including the lid-driven cavity flow and the square cylinder flow at high Reynolds numbers, and then, we applied the method to the LES of turbulent flows around a full-size high-speed train with realistic geometry and operating conditions. The numerical results showed that the algorithm is both accurate and efficient and exhibits a good scalability and parallel efficiency with up to 4096 processors. We showed numerically that, by using the fully implicit method the time step size is no longer constrained by the CFL condition so that large time step sizes are allowed. We studied how the algorithm is influenced by the choices of linear solvers, the overlapping size of the subdomains, and, especially, the accuracy of the Jacobian matrix. The results show that an exact Jacobian is necessary for the efficiency and the robustness of the proposed LES solver.

ACKNOWLEDGEMENTS

This research was supported in part by the National Key R&D Program of China under grant 2016YFB0200601; by the Shenzhen Basic Research Program under grants JCYJ20160331193229720, JCYJ20170307165328836, and JSGG20170824154458183; and by the National Natural Science Foundation of China under grants 61531166003, 11602282, and DMS-1720366.

ORCID

Xiao-Chuan Cai  <https://orcid.org/0000-0003-0296-8640>

REFERENCES

1. Gokarn A, Battaglia F, Fox RO, Hill JC, Reveillon J. Large eddy simulations of incompressible turbulent flows using parallel computing techniques. *Int J Numer Methods Fluids*. 2008;56(10):1819-1843.
2. Hsu H-W, Hwang F-N, Wei Z-H, Lai S-H, Lin C-A. A parallel multilevel preconditioned iterative pressure poisson solver for the large-eddy simulation of turbulent flow inside a duct. *Comput Fluids*. 2011;45(1):138-146.
3. Colomés O, Badia S. Segregated Runge–Kutta time integration of convection-stabilized mixed finite element schemes for wall-unresolved LES of incompressible flows. *Comput Meth Appl Mech Eng*. 2017;313:189-215.
4. Situ Y, Martha CS, Louis ME, et al. Petascale large eddy simulation of jet engine noise based on the truncated SPIKE algorithm. *Parallel Comput*. 2014;40(9):496-511.
5. Singh KM, Avital EJ, Williams JJR, Ji C, Bai X, Munjiza A. On parallel pre-conditioners for pressure poisson equation in LES of complex geometry flows. *Int J Numer Methods Fluids*. 2017;83(5):446-464.
6. Antepara O, Lehmkuhl O, Borrell R, Chiva J, Oliva A. Parallel adaptive mesh refinement for large-eddy simulations of turbulent flows. *Comput Fluids*. 2015;110:48-61.
7. Su M, Yu J-D. A parallel large eddy simulation with unstructured meshes applied to turbulent flow around car side mirror. *Comput Fluids*. 2012;55:24-28.
8. Forti D, Dedè L. Semi-implicit BDF time discretization of the Navier–Stokes equations with VMS–LES modeling in a high performance computing framework. *Comput Fluids*. 2015;117:168-182.
9. Moureau V, Domingo P, Vervisch L. Design of a massively parallel CFD code for complex geometries. *Comptes Rendus Mécanique*. 2011;339(2-3):141-148.
10. Nichols J, Lele S, Moin P, Ham F, Bridges J, Brés J. Large-eddy simulation for supersonic rectangular jet noise prediction: effects of chevrons. Paper presented at: 18th AIAA/CEAS Aeroacoustics Conference (33rd AIAA Aeroacoustics Conference); 2012; Colorado Springs, CO.
11. Posey S, Loewe B, Calleja P. Cluster scalability of ANSYS FLUENT 12 for a large aerodynamics case on the Darwin Supercomputer. Paper presented at: 4th European Automotive Simulation Conference; 2009; Munich, Germany.
12. Prudencio EE, Byrd R, Cai X-C. Parallel full space SQP Lagrange–Newton–Krylov–Schwarz algorithms for PDE-constrained optimization problems. *SIAM J Sci Comput*. 2006;27(4):1305-1328.
13. Barker AT, Cai X-C. Scalable parallel methods for monolithic coupling in fluid–structure interaction with application to blood flow modeling. *J Comput Phys*. 2010;229(3):642-659.
14. Wu Y, Cai X-C. A fully implicit domain decomposition based ALE framework for three-dimensional fluid–structure interaction with application in blood flow computation. *J Comput Phys*. 2014;258:524-537.
15. Shiu W-S, Hwang F-N, Cai X-C. Parallel domain decomposition method for finite element approximation of 3D steady state non-Newtonian fluids. *Int J Numer Methods Fluids*. 2015;78(8):502-520.
16. Deng X, Cai X-C, Zou J. Two-level space–time domain decomposition methods for three-dimensional unsteady inverse source problems. *J Sci Comput*. 2016;67(3):860-882.
17. Kong F, Cai X-C. 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*. 2016;38(2):C73-C95.
18. Smagorinsky J. General circulation experiments with the primitive equations: I. the basic experiment. *Mon Weather Rev*. 1963;91(3):99-164.
19. Sagaut P. *Large Eddy Simulation for Incompressible Flows: An Introduction*. Berlin, Germany: Springer-Verlag Berlin Heidelberg; 2006.
20. Germano M, Piomelli U, Moin P, Cabot WH. A dynamic subgrid-scale eddy viscosity model. *Phys Fluids A Fluid Dyn*. 1991;3(7):1760-1765.
21. Lilly DK. A proposed modification of the Germano subgrid-scale closure method. *Phys Fluids A Fluid Dyn*. 1992;4(3):633-635.
22. Franca LP, Frey SL. Stabilized finite element methods: II. The incompressible Navier-Stokes equations. *Comput Meth Appl Mech Eng*. 1992;99(2-3):209-233.
23. Whiting CH, Jansen KE. A stabilized finite element method for the incompressible Navier-Stokes equations using a hierarchical basis. *Int J Numer Methods Fluids*. 2001;35(1):93-116.
24. Hu Q, Zou J. Nonlinear inexact Uzawa algorithms for linear and nonlinear saddle-point problems. *SIAM J Optim*. 2006;16(3):798-825.
25. Eisenstat SC, Walker HF. Globally convergent inexact Newton methods. *SIAM J Optim*. 1994;4(2):393-422.
26. Eisenstat SC, Walker HF. Choosing the forcing terms in an inexact Newton method. *SIAM J Sci Comput*. 1996;17(1):16-32.
27. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J Sci Stat Comput*. 1986;7(3):856-869.
28. Cai X-C, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J Sci Comput*. 1999;21(2):792-797.
29. Saad Y. *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2003.
30. Balay S, Abhyankar S, Adams MF, et al. *PETSc Users Manual*. Technical report. Chicago, IL: Argonne National Laboratory; 2016. ANL-95/11 Rev 3.7.
31. Prasad AK, Koseff JR. Reynolds number and end-wall effects on a lid-driven cavity flow. *Phys Fluids A Fluid Dyn*. 1989;1(2):208-218.
32. Padilla ELM, Martins AL, Silveira-Neto A. Large-eddy simulation of the three-dimensional unstable flow in a lid-driven cavity. Paper presented at: 18th International Congress of Mechanical Engineering; 2005; Ouro Preto, Brazil.
33. Rodi W, Ferziger JH, Breuer M, Pourquie M. Status of large eddy simulation: results of a workshop. *J Fluids Eng*. 1997;119(2):248-262.

34. Sohankar A, Davidson L, Norberg C. Large eddy simulation of flow past a square cylinder: comparison of different subgrid scale models. *J Fluids Eng.* 2000;122(1):39-47.
35. Koobus B, Farhat C. A variational multiscale method for the large eddy simulation of compressible turbulent flows on unstructured meshes—application to vortex shedding. *Comput Meth Appl Mech Eng.* 2004;193(15-16):1367-1383.
36. Lyn DA, Einav S, Rodi W, Park J-H. A laser-Doppler velocimetry study of ensemble-averaged characteristics of the turbulent near wake of a square cylinder. *J Fluid Mech.* 1995;304:285-319.
37. Bearman PW, Obasaju ED. An experimental study of pressure fluctuations on fixed and oscillating square-section cylinders. *J Fluid Mech.* 1982;119:297-321.
38. Kogaki T, Kobayashi T, Taniguchi N. Large eddy simulation of flow around a rectangular cylinder. *Fluid Dyn Res.* 1997;20(1-6):11-24.

How to cite this article: Liao Z-J, Chen R, Yan Z, Cai X-C. A parallel implicit domain decomposition algorithm for the large eddy simulation of incompressible turbulent flows on 3D unstructured meshes. *Int J Numer Meth Fluids.* 2019;89:343–361. <https://doi.org/10.1002/flid.4695>