Noname manuscript No.
(will be inserted by the editor)

A Parallel Domain Decomposition Method for 3D Unsteady Incompressible Flows at High Reynolds Number

Rongliang Chen · Yuqi Wu · Zhengzheng Yan · Yubo Zhao · Xiao-Chuan Cai

the date of receipt and acceptance should be inserted later

Abstract Direct numerical simulation of three-dimensional incompressible flows at high Reynolds number using the unsteady Navier-Stokes equations is challenging. In order to obtain accurate simulations, very fine meshes are necessary, and such simulations are increasingly important for modern engineering practices, such as understanding the flow behavior around high speed trains, which is the target application of this research. To avoid the time step size constraint imposed by the CFL number and the fine spacial mesh size, we investigate some fully implicit methods, and focus on how to solve the large nonlinear system of equations at each time step on large scale parallel computers. In most of the existing implicit Navier-Stokes solvers, segregated velocity and pressure treatment is employed. In this paper, we focus on the Newton-Krylov-Schwarz method for solving the monolithic nonlinear system arising from the fully coupled finite element discretization of the Navier-Stokes equations on unstructured meshes. In the subdomain, LU or point-block ILU is used as the local solver. We test the algorithm for some three-dimensional complex unsteady flows, including flows passing a high speed train, on a supercomputer with thousands of processors. Numerical experiments show that the algorithm has superlinear scalability with over three thousand processors for problems with tens of millions of unknowns.

The research was supported in part by NSF of USA under grants DMS-0913089 and CCF-1216314, the Knowledge Innovation Program of the Chinese Academy of Sciences (China) under KJCX2-EW-L01, and the international cooperation project of Guangdong provience (China) under 2011B050400037.

R. Chen · Z. Yan · Y. Zhao
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong 518055, China
E-mail: rl.chen@siat.ac.cn, zz.yan@siat.ac.cn, yb.zhao@siat.ac.cn

Y. Wu
Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA
E-mail: yuqiwu@uw.edu

X.-C. Cai
Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA
E-mail: cai@cs.colorado.edu

Keywords Three-dimensional unsteady incompressible flows · direct numerical simulation · parallel computing · fully implicit · domain decomposition

Mathematics Subject Classification (2000) 76D05 · 76F65 · 65M55 · 65Y05

1 Introduction

Because of the advancement of supercomputing, parallel computational fluid dynamics has become an enabling technology supporting a wide range of applications in science and engineering. For example, simulations of flow around an object with a complicated shape are particularly useful in aerodynamic vehicle design. Many engineering and atmospheric flows are turbulent, and understanding the behavior of the flows is of great practical importance. Roughly speaking, turbulence simulation methodologies can be classified into Reynolds-averaged Navier-Stokes approaches (RANS), large-eddy simulation (LES), and direct numerical simulation (DNS). RANS is based on the Reynolds decomposition of the flow variables into their time-averaged and fluctuating quantities. We refer interested readers to [1] for details. LES is intermediate between RANS and DNS, which ignores the small turbulent scales and computes the dynamics of the large scales [26]. LES was the most popular technique for the turbulence simulations in the last few decades [13, 18, 22]. DNS is the most accurate method for the numerical simulation of turbulent flows, and is also the most expensive one in terms of the computational cost. In DNS, the Navier-Stokes equations are numerically solved without any turbulence model, which means that the momentum equation of the Navier-Stokes equations must be exactly solved. DNS is an useful tool in fundamental research on turbulence, and using DNS it is possible to perform certain "experiments" that are difficult or sometimes impossible to obtain in actual experiments. Some reviews about the DNS can be found in the references [2, 12, 19]. In this study, we focus on studying 3D incompressible flows around complex bodies. A key motivation for the current work is to simulate unsteady realistic flow around a high speed train. Because of the lack of parallel scalability, commercial CFD software packages can only be used when the mesh is not too fine, and thus don't offer sufficient accuracy.

In DNS, in order to obtain sufficiently accurate solutions, small spatial scales of the complex flows must be resolved by the computational mesh, thus the computation is usually very demanding, requiring large scale parallel computers for their memory capacity and processing speed. It is clear by now that the increase of computing power is no longer from faster processors, but from the increase of the number of processors. This makes the scalability of the algorithm more important than ever. Many researchers have studied parallel algorithms for DNS. For examples, Yokokawa et al. [29] studied DNS of incompressible turbulence flows with the Fourier spectral method using over 4000 processors; Chen [9] studied DNS of chemistry turbulence on a Petascale supercomputer using a finite difference method for the spatial discretization and an explicit Runge-Kutta method for the temporal discretization; Rahimian et al. [23] studied DNS of blood flows using nearly 200 thousand processor cores with over 90 billion unknowns.

In this work, we present a Newton-Krylov-Schwarz (NKS) based parallel implicit solver for the unsteady incompressible Navier-Stokes equations. NKS is a general purpose parallel solver for nonlinear systems and has been widely used to

solve different kind of nonlinear problems; see e.g., [6, 14, 15, 20]. Our algorithm begins with a discretization of the incompressible unsteady Navier-Stokes equations on an unstructured tetrahedron mesh with a stabilized finite element method in space and a fully implicit backward difference scheme in time. At each time step, an inexact Newton method is employed to solve the discretized large sparse nonlinear system and in the Newton steps, a domain decomposition preconditioned Krylov method is used to solve the Jacobian system which is constructed analytically in order to obtain the desired performance. The most important component of the solver is the monolithic Schwarz preconditioner that keeps the strong coupling of the velocity and pressure variables at each mesh point throughout the entire algorithm.

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

2 Governing equations

The Navier-Stokes equations are the fundamental governing equations that describe the flow of a viscous fluid. In this paper, the incompressible unsteady Navier-Stokes equations are used to model the flow. Let $\Omega \subset R^3$ be the spatial domain of interest bounded by the boundary $\Gamma = \Gamma_{inlet} \cup \Gamma_{wall} \cup \Gamma_{outlet}$. The equations read as, in the vector form:

$$\begin{aligned} \rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) - \nabla\cdot\sigma &= \mathbf{f} \text{ in } \Omega, \\ \nabla\cdot\mathbf{u} &= 0 \text{ in } \Omega, \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the velocity, $\sigma = -p\mathbf{I} + \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$ is the Cauchy stress tensor, p is the pressure, ρ is the fluid density, μ is the dynamic viscosity, and $\mathbf{f}$ represents the body force or the source term. A given velocity profile $\mathbf{g}$ is chosen on the inlet boundary Γ_{inlet}, no-slip boundary conditions are used on the wall Γ_{wall} and on the outlet boundary Γ_{outlet} the stress-free boundary conditions are imposed:

$$\begin{aligned} \mathbf{u} &= \mathbf{g} \quad \text{on} \quad \Gamma_{inlet}, \\ \mathbf{u} &= \mathbf{0} \quad \text{on} \quad \Gamma_{wall}, \\ \sigma\cdot\mathbf{n} &= \mathbf{0} \quad \text{on} \quad \Gamma_{outlet}, \end{aligned} \tag{2}$$

where $\mathbf{n}$ is the outward unit normal vector on the domain boundary. The initial condition for the velocity is specified as:

$$\mathbf{u} = \mathbf{u_0} \quad \text{in} \quad \Omega \quad \text{at} \quad t = 0. \tag{3}$$

Here $\mathbf{u_0}$ is a given function.

3 Fully-implicit finite element discretization

We use a $P1-P1$ finite element method to discretize the Navier-Stokes equations (1) in the spatial domain. To describe the finite element method, we first define the trial and weighting function spaces as

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

Then, the weak form of the Navier-Stokes equations takes the form: Find $\mathbf{u} \in \mathcal{U}$, $p \in \mathcal{P}$ such that

$$\begin{aligned}
&\rho \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \Phi d\Omega + \mu \int_\Omega \nabla \mathbf{u} : \nabla \Phi d\Omega + \rho \int_\Omega (\mathbf{u} \cdot \nabla)\mathbf{u} \cdot \Phi d\Omega \\
&\qquad - \int_\Omega p \nabla \cdot \Phi d\Omega + \int_\Omega (\nabla \cdot \mathbf{u}) \varphi d\Omega = \int_\Omega \mathbf{f} \cdot \Phi d\Omega,
\end{aligned} \tag{4}$$

holds for all $\Phi \in \mathcal{U}_0$ and $\varphi \in \mathcal{P}$.

The finite element discretization begins with meshing the computational domain with an unstructured tetrahedral mesh $\mathcal{T}^h = \{K\}$. The finite dimensional trial and weighting spaces can then be established as

$$\begin{aligned}
\mathcal{U}^h &= \{\mathbf{u}^h(\cdot,t) \mid \mathbf{u}^h(\cdot,t) = \sum_{i=1}^{N_u} \Phi_i^h \mathbf{u}_i^h(\cdot,t), \quad \mathbf{u}^h(\cdot,t) = \mathbf{g} \quad \text{on} \quad \Gamma_{inlet}\},\\
\mathcal{U}_0^h &= \{\mathbf{u}^h(\cdot,t) \mid \mathbf{u}^h(\cdot,t) = \sum_{i=1}^{N_u} \Phi_i^h \mathbf{u}_i^h(\cdot,t), \quad \mathbf{u}^h(\cdot,t) = \mathbf{0} \quad \text{on} \quad \Gamma_{inlet} \cup \Gamma_{wall}\},\\
\mathcal{P}^h &= \{p^h(\cdot,t) \mid p^h(\cdot,t) = \sum_{i=1}^{N_p} \varphi_i^h p_i^h(\cdot,t)\},
\end{aligned}$$

where $\mathbf{u}_i^h \in R^3$, $p_i^h \in R$ are the nodal values of the velocity and pressure functions. N_u and N_p are the number of nodes for velocity and pressure, respectively. Each of the three components of Φ_i^h and φ_i^h are the basis functions which are piecewise linear functions. Since the $P1-P1$ element does not satisfy the Ladyzenskaja-Babuska-Brezzi (LBB) condition, we need to add suitable stabilization terms to fulfill the LBB condition. For this purpose, we employ the stabilization technique introduced in publications [5,11]. The semi-discrete stabilized finite element formulation of (4) is given as follows: Find $\mathbf{u}^h \in \mathcal{U}^h$, $p \in \mathcal{P}^h$ such that

$$\begin{aligned}
&\rho \int_\Omega \frac{\partial \mathbf{u}^h}{\partial t} \cdot \Phi^h d\Omega + \mu \int_\Omega \nabla \mathbf{u}^h : \nabla \Phi^h d\Omega + \rho \int_\Omega (\mathbf{u}^h \cdot \nabla)\mathbf{u}^h \cdot \Phi^h d\Omega \\
&- \int_\Omega p^h \nabla \cdot \Phi^h d\Omega + \int_\Omega (\nabla \cdot \mathbf{u}^h) \varphi^h d\Omega + \underline{\sum_{K \in \mathcal{T}^h} \left(\nabla \cdot \mathbf{u}^h,\ \tau_c \nabla \cdot \Phi^h\right)_K} \\
&\underline{\sum_{K \in \mathcal{T}^h} \left(\frac{\partial \mathbf{u}^h}{\partial t} + (\mathbf{u}^h \cdot \nabla)\mathbf{u}^h + \nabla p^h,\ \tau_m(\mathbf{u}^h \cdot \nabla \Phi^h + \nabla \varphi^h)\right)_K} \\
&= \int_\Omega \mathbf{f} \cdot \Phi^h d\Omega + \underline{\sum_{K \in \mathcal{T}^h} \left(\mathbf{f},\ \tau_m(\mathbf{u}^h \cdot \nabla \Phi^h + \nabla \varphi^h)\right)_K},
\end{aligned} \tag{5}$$

holds for all $\Phi^h \in \mathcal{U}_0^h$ and $\varphi^h \in \mathcal{P}^h$. The underlined terms are the stabilization terms where the parameters τ_c and τ_m are defined as follows:

$$\tau_m = \left(\sqrt{\frac{4}{\Delta t^2} + (\mathbf{u} \cdot G\mathbf{u}) + 36 \left(\frac{\mu}{\rho}\right)^2 G : G} \right)^{-1},$$
$$\tau_c = \frac{1}{8\tau_m tr(G)}.$$

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 and $\frac{\partial \xi}{\partial x}$ represents the Jacobian of the mapping between the reference and the physical element.

We use an implicit backward finite difference formula with a fixed time step size, Δt, to discretize (5) in time. For a given semi-discretized system

$$\frac{dx}{dt} = L(x),$$

the formula is defined as

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

At the n^{th} time step, we need to solve a nonlinear system

$$\mathbf{F}^n(X^n) = \mathbf{0}, \tag{6}$$

with the initial guess X^{n-1} (the solution of the $(n-1)^{th}$ time step), to obtain the solution of the n^{th} time step X^n, which is the nodal values of the velocity and pressure. The ordering of the nodal values and the corresponding nonlinear functions is not important for the accuracy of the solution, but is very important for the convergence of the algebraic solver and also the performance of the solver in terms of the computing time. In most existing approaches, the field-by-field ordering is often used, as a result, the Jacobian of the nonlinear system has a saddle point structure, which plays a key role in the design of the iterative method and its preconditioner. We do not use the field-by-field ordering. We order the variables and functions element by element and in each element, the variables are ordered node by node. This ordering helps constructing the point-block incomplete LU factorization that is more stable than the classical pointwise ILU, and also improving the cache performance and the parallel efficiency in load and communication balance.

4 Monolithic Newton-Krylov-Schwarz algorithm

In most Navier-Stokes solvers, such as the projection methods, the operator is split into the velocity component and pressure component, and the algorithm takes the form of a nonlinear Gauss-Seidel iteration with two large blocks. In the monolithic approach that we consider in this paper, the velocity and pressure variables of a grid point stay together throughout the computation. In this approach, the two critical ingredients are the monolithic Schwarz preconditioner, and the point-block ILU based subdomain solver.

The nonlinear system (6) is solved by a Newton-Krylov-Schwarz method which uses an inexact Newton method [10] as the nonlinear solver, a Krylov subspace

method (GMRES) [25] as the linear solver at each Newton step, and an overlapping Schwarz method [7] as the preconditioner. The framework of the Newton-Krylov-Schwarz method reads as

Algorithm NKS

Step 1. *Use the solution of the previous time step as the initial guess* $\mathbf{X}_0^n = \mathbf{X}^{n-1}$

Step 2. *For* $k = 0, 1, \cdots$ *until convergence*

- *Construct the complete Jacobian matrix* $\mathbf{J}_k^n$
- *Solve the following right-preconditioned Jacobian system inexactly by a Krylov subspace method*

$$\mathbf{J}_k^n (\mathbf{M}_k^n)^{-1} \mathbf{M}_k^n \mathbf{S}_k^n = -\mathbf{F}^n(\mathbf{X}_k^n) \tag{7}$$

- *Do a cubic line search to find a step length* τ_k^n
- *Set* $\mathbf{X}_{k+1}^n = \mathbf{X}_k^n + \tau_k^n \mathbf{S}_k^n$

Here $\mathbf{J}_k^n$ is the full Jacobian of $\mathbf{F}^n(\mathbf{X})$ at point $\mathbf{X}_k^n$, including the stabilization terms, $\mathbf{M}_k^n$ is an additive Schwarz preconditioner to be introduced shortly. The inexactness mentioned in Step 2 means that the accuracy of the solution to the Jacobian system (7) is in the sense of

$$\| \mathbf{J}_k^n (\mathbf{M}_k^n)^{-1} \mathbf{M}_k^n \mathbf{S}_k^n + \mathbf{F}^n(\mathbf{X}_k^n) \| \leq \eta_k^n \| \mathbf{F}^n(\mathbf{X}_k^n) \|, \tag{8}$$

where η_k^n is the relative tolerance for the linear solver. For simplicity, we ignore the scripts n and k for the rest of the paper.

In NKS, the most difficult and time-consuming step is the solution of the large, sparse, and nonsymmetric Jacobian system (7) by a preconditioned GMRES method. In the Jacobian solver, the most important component is the preconditioner; without which GMRES method doesn't converge or converges very slowly, and a good choice of preconditioner accelerates the convergence significantly. Let n_p be the number of processors of the parallel machine. In this paper, we use an overlapping restricted additive Schwarz preconditioner [7], where we first partition the computational domain Ω into n_p nonoverlapping subdomains Ω_l $(l = 1, \cdots, n_p)$ and then extend each subdomain Ω_l to an overlapping subdomain Ω_l^δ by including δ layers of elements belonging to its neighbors. In each overlapping subdomain, we define a local Jacobian matrix $\mathbf{J}_l$ which is the restriction of the global Jacobian matrix $\mathbf{J}$ to Ω_l^δ with the restriction operator R_l^δ. R_l^δ is a matrix which maps the global vector of unknowns to those belonging to Ω_l^δ by simply extracting the unknowns that lie inside the subdomain. In practice, $\mathbf{J}_l$ is obtained by taking the derivatives of the discretized Navier-Stokes equations (1) in Ω_l^δ with homogeneous Dirichlet boundary conditions in the interior of Ω, and the physical boundary conditions on $\partial\Omega$. The restricted additive Schwarz preconditioner is defined as the summation of the local preconditioners $\mathbf{B}_l^{-1}$ of $\mathbf{J}_l$:

$$\mathbf{M}_{RAS} = \sum_{l=1}^{n_p} (R_l^0)^T \mathbf{B}_l^{-1} R_l^\delta, \tag{9}$$

where the restriction operator R_l^0 is defined as the restriction to the unknowns in the non-overlapping subdomain Ω_l. In practice, we only need the application

of $\mathbf{B}_l^{-1}$ to a given vector, which can be obtained by solving a subdomain linear system. Since $\mathbf{B}_l^{-1}$ is used as a preconditioner here, the subdomain linear system can be solved exactly or approximately. Both approaches are studied in this paper. LU factorization is computationally expensive and requires large memory resources when the local matrix $\mathbf{J}_l$ is large. An economical alternative is the incomplete LU factorization (ILU) [8, 24] which reduces the computation by dropping some fill-in elements in predetermined nondiagonal positions that are generated during the factorization process. In this paper, we use a point-block ILU as the local preconditioner, where we group all physical components associated with a mesh point as a block and always perform an exact LU factorization for this small block, in addition, the velocity and pressure variables associated with a given mesh point is either kept or dropped together. The effectiveness and the computational cost of the subdomain preconditioner depend on the number of elements dropped.

5 Numerical experiments

In this section, we report some numerical results of the proposed algorithm. Our solver is implemented on top of the Portable Extensible Toolkit for Scientific computation (PETSc) [3]. Even though, most components of our discretization scheme have been studied by others, but the overall finite element scheme is new. To test its correctness, we first simulation a flow around a cylinder. The second test case represents our target application. The unstructured tetrahedral meshes for the first test case are generated with CUBIT [21] from Sandia National Laboratory and the geometry of the second test case is generated with AutoCAD and meshed by using ANSYS. The mesh partitions for the additive Schwarz preconditioner are obtained with ParMETIS [17] of University of Minnesota. The results showed in this section are obtained on the Dell PowerEdge C6100 Cluster at the University of Colorado Boulder. The stopping conditions for the nonlinear and linear solver are that when the residuals of the nonlinear and linear equations are reduced by a factor of 10^{-12} and 10^{-6}, respectively.

In the simulations, a very important parameter is the Reynolds number (Re) which is a dimensionless number that determines the ratio of inertial forces to viscous forces. Usually, a low Reynolds number implies a laminar flow and a high Reynolds number corresponds to a turbulent flow. The Reynolds number is defined as

$$Re = \frac{\rho \overline{\mathbf{u}} L}{\mu}, \tag{10}$$

where $\overline{\mathbf{u}}$ is the mean velocity of the object relative to the fluid. In this paper, we choose $\overline{\mathbf{u}}$ as the mean velocity at the inlet boundary. L is the characteristic length and it is the diameter of the obstacle in this paper.

5.1 Benchmark problem

We first test the algorithm for a well-understood benchmark problem, flow around a cylinder, defined in [4, 27]. The detailed geometry of the problem is shown in Fig. 1. As suggested in [27], two important features of this flow are the drag and

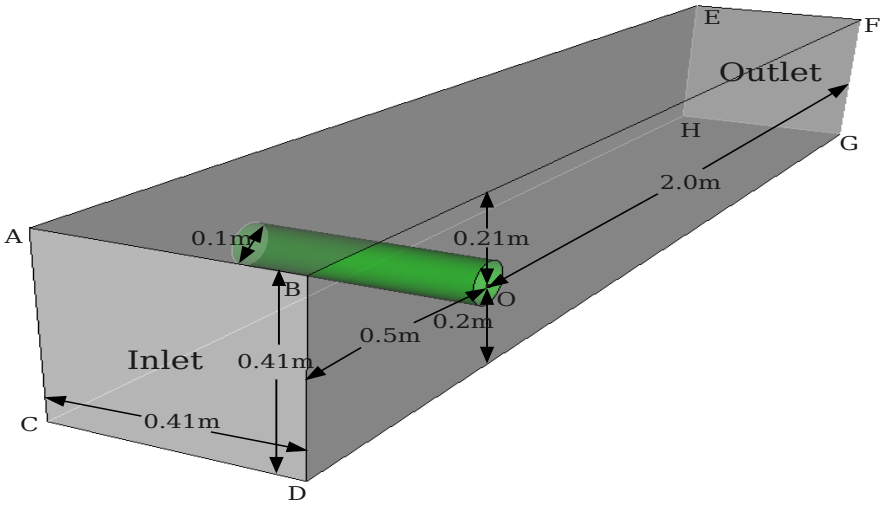

Fig. 1 Flow passing a cylinder in a channel

the lift coefficient of the cylinder. The definitions of these two coefficients read as

$$Drag = \frac{2F_d}{\rho U_m^2 DH} \quad \text{and} \quad Lift = \frac{2F_l}{\rho U_m^2 DH}, \tag{11}$$

respectively, where F_d and F_l are defined as

$$F_d = \int_S \left(\rho\mu \frac{\partial u_t}{\partial n} n_y - p n_x\right) dS \quad \text{and} \quad F_l = -\int_S \left(\rho\mu \frac{\partial u_t}{\partial n} n_x - p n_y\right) dS. \tag{12}$$

Here S is the surface of the cylinder, $H = 0.41m$ is the height of the channel, $D = 0.1m$ is the diameter of the cylinder, n_x and n_y are the normal vectors, u_t is the tangential velocity of $\mathbf{u}$ with $t = (n_y, -n_x, 0)$ and U_m is the maximal inflow $\mathbf{u}_{in} = (u_{in}, v_{in}, w_{in})$ ([27]) with

$$u_{in} = 36 \sin\left(\frac{\pi t}{8}\right) \frac{yz(H-y)(H-z)}{H^4}, \quad v_{in} = w_{in} = 0. \tag{13}$$

In this test case, the kinematic viscosity $\mu = 10^{-3} m^2/s$ and the density $\rho = 1 kg/m^3$. The drag and lift coefficients of the cylinder are shown in Fig. 2. These results are obtained on a mesh with about 1.6×10^7 elements (total degrees of

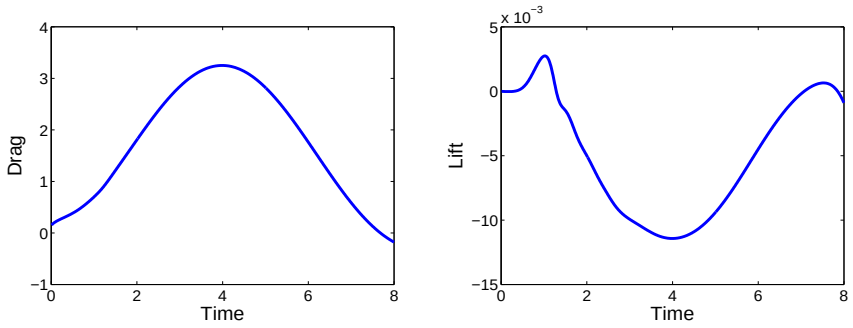

Fig. 2 Drag coefficient (left) and lift coefficient (right) for the laminar flow.

Table 1 Number of iterations and total compute time for a laminar flow on a mesh with 1.6×10^7 elements (mesh size $h \approx 0.001$) using different time step sizes. The computations are carried with 2400 processors.

Δt	Newton	GMRES	Time
0.01	3.0	102.2	106.5
0.05	3.0	114.6	137.2
0.1	3.0	120.0	112.0
0.5	4.1	137.3	179.7
1.0	4.9	147.5	241.7

freedom $DOF = 1.1 \times 10^7$) and a fixed time step $\Delta t = 0.01s$ with the zero initial condition and zero body force. The maximal drag coefficient $Drag_{max} = 3.2507$, minimal lift coefficient $Lift_{min} = -0.011427$ and maximal lift coefficient $Lift_{max} = 0.002744$. Another important parameter to compute is the difference of the pressure at the final time $t = 8s$ between the front and back of the cylinder, and the value is -0.1131 in our test case and it agrees well with the results of [16, 27].

We next study the numerical performance of the algorithm. Since we use a fully implicit method in which the time step size is no longer constrained by the Courant-Friedrichs-Lewy (CFL) condition, we can use very large time step size. Table 1 shows the number of linear and nonlinear iterations and the compute time of the fully implicit method with respect to different time step sizes. From this table, we see that the algorithm is stable, and converges quite well with different time step size. The parallel performance of the algorithm is shown in Table 2. This table shows that when we increase the number of processors, the average number of Newton iterations (Newton) per time step does not change, the average number of GMRES iterations per Newton step (GMRES) increases reasonably, and the compute time per time step decreases quickly. The left figure of Fig. 3 shows the speedup of the algorithm for this problem and it indicates that the algorithm has a superlinear speedup. The blue line refers to the linear speedup which means that the compute time is exactly halved when the number of processors is doubled, and the red line is the actual speedup of the algorithm. The right figure of Fig. 3 is the average compute time per time step with respect to the number of processors. We note that the number of Newton iterations is small in the test cases. This is

Table 2 Parallel performance of the algorithm for the laminar flow simulation. Here $DOF = 1.1 \times 10^7$ and $Re = 20$.

n_p	Newton	GMRES	Time
1024	3.1	61.5	597.5
1536	3.1	66.6	272.0
2048	3.1	80.3	180.1
3072	3.1	105.4	101.6

because the full Jacobian is used in the algorithm. If the derivative with respect to some of the stabilization terms are dropped, the number of Newton iterations increases.

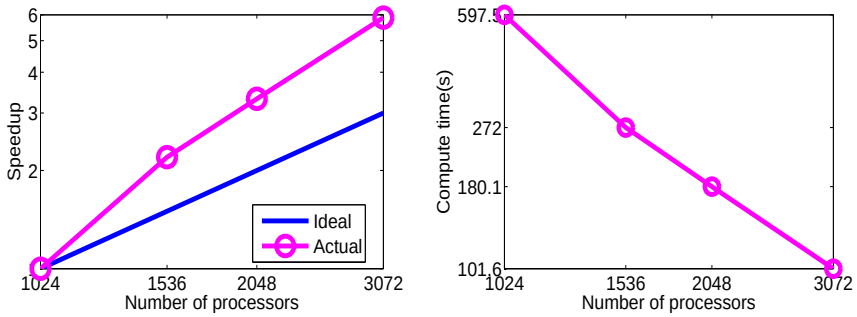

Fig. 3 The speedup and the average compute time per time step (log-log scaled) for the flow passing a cylinder simulation. Here $DOF = 1.1 \times 10^7$ and $Re = 20$.

5.2 High speed train simulation

In this section, we study a flow passing a high speed train with a realistic train geometry, and a realistic Reynolds number. Many engineering and safety problems are being raised with the rapid development of high speed rail transportation. The wind conditions around a train body influence the stability of the train significantly. A thorough understanding of the wind around the train is extremely important in the shape design of the train, and also in the control of the train under different weather conditions, etc. The computation of this 3D problem is very demanding because of the large computational domain, the complexity of the geometry and the ill-conditionness of the discretized mathematical model.

A realistic three dimensional train model with two cars is considered in this paper. The geometry is created by AutoCAD and the flow domain is meshed by ANSYS; see Fig. 4 for the details of the train model, the computational domain and a local view of the computational mesh. Standard flow parameters include the dynamic viscosity $\mu = 1.831 \times 10^{-5} kg/(m \cdot s)$ and the density $\rho = 1.185 kg/m^3$. The boundary conditions are defined as follows: a time-varying boundary condition $\mathbf{u}_{in} = (0, v_{in} \cdot t, 0)$ is employed on the inlet Γ_{inlet} (when $v_{in} \cdot t > 100$, let $\mathbf{u}_{in} = (0, 100, 0)$) where v_{in} is a constant and t is a time, stress free boundary condition

($\sigma \cdot \mathbf{n} = \mathbf{0}$) is used on the outlet boundary Γ_{outlet} and a no-slip boundary condition ($\mathbf{u} = \mathbf{0}$) is given on the wall boundary Γ_{wall} (all the surfaces except Γ_{inlet} and Γ_{outlet}). The zero initial condition and zero body force are used for this test case. We assume the velocity of the train is $360km/h$, that is, $v_{in} = 100m/s$ for the inlet boundary condition. The Reynolds number for this test case

$$Re = \frac{\rho v_{in} L}{\mu} = \frac{1.185kg/m^3 \cdot 100m/s \cdot 3m}{1.831 \times 10^{-5} kg/(m \cdot s)} = 1.9 \times 10^7,$$

where the characteristic length L is chosen as the height of the train.

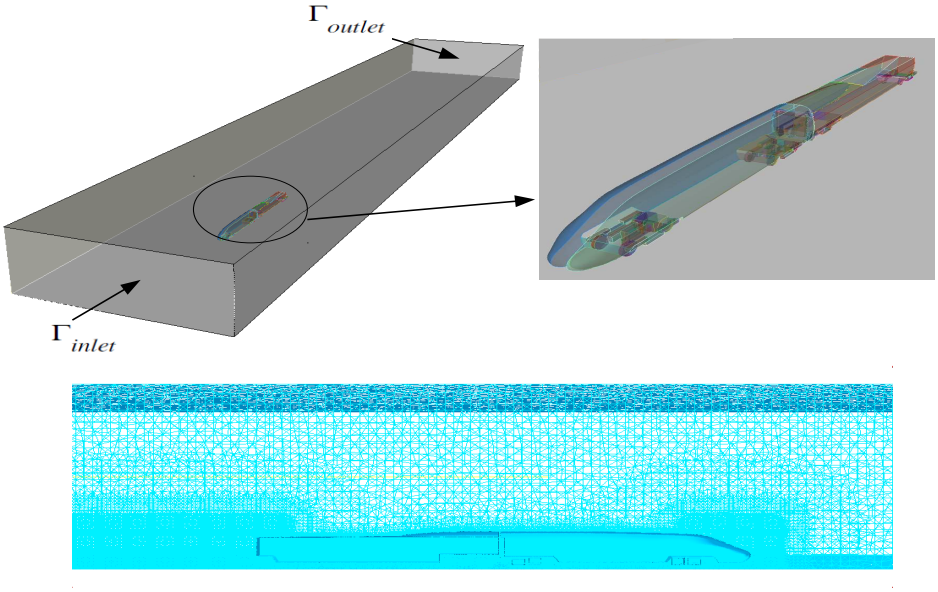

Fig. 4 Model for the simulation of the flow around a high speed train (top) and the mesh around the train (bottom). Here the size of the box (top left) is $140m \times 23m \times 9m$ and the dimension of the train (top right) is $19m \times 3m \times 2m$.

The velocity magnitudes distribution around the train at $t = 1.0s$ (time step size $\Delta t = 0.01$) is shown in Fig. 5 and Fig. 6. From these figures, we see that the flow is very complicated at the end of the train and under the train, and more details can be viewed in the stream trace figures Fig. 7 and Fig. 8.

To investigate the parallel performance and parallel scalability of the algorithm for this complicated problem, we choose two meshes, one with about 1.1×10^7 elements (8 million degrees of freedom (DOF)), and the other with about 2.2×10^7

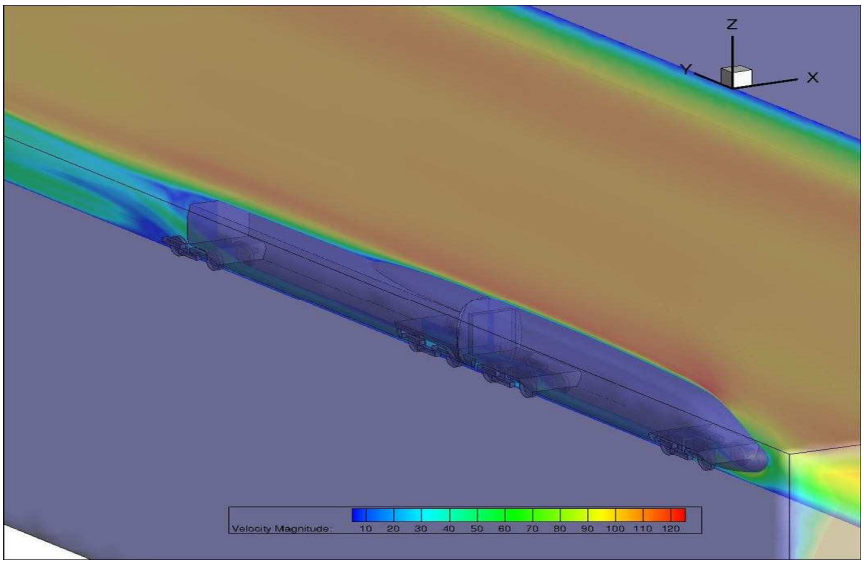

Fig. 5 The velocity distribution around the train at $t = 1.0s$. Here $v_{in} = 100m/s$, $\mu = 1.83 \times 10^{-5} pa \cdot s$, $Re = 10^7$.

Table 3 Parallel performance of the high speed train simulation. Here $\delta = 6$ and $Re = 10^7$.

n_p	$DOF = 8 \times 10^6$			$DOF = 1.7 \times 10^7$		
	Newton	GMRES	Time	Newton	GMRES	Time
1024	4.0	91.4	640.3	4.0	55.3	1501.2
1536	4.0	113.5	373.1	4.0	59.6	823.0
2048	4.0	136.2	266.8	4.0	62.1	420.6
3072	4.0	152.6	173.7	4.0	66.3	273.1

elements (17 million DOF). Table 3 shows the average number of Newton iterations per time step, the average number of GMRES iterations per Newton step and the average compute time per time step. The results are averaged values over the first 10 time steps. From this table, we see that the number of Newton iterations does not change and the average number of GMRES iterations increases mildly as the number of processors increases. The compute time is more than halved as the number of processors is doubled, which means that the algorithm has a superlinear speedup. The superlinear speedup is also shown in the left figure of Fig. 9. The right figure of Fig. 9 is the average compute time per time step with respect to the number of processors.

In these experiments just shown, the subdomain problems are solved by LU factorization which is expensive in both computation and memory requirement.

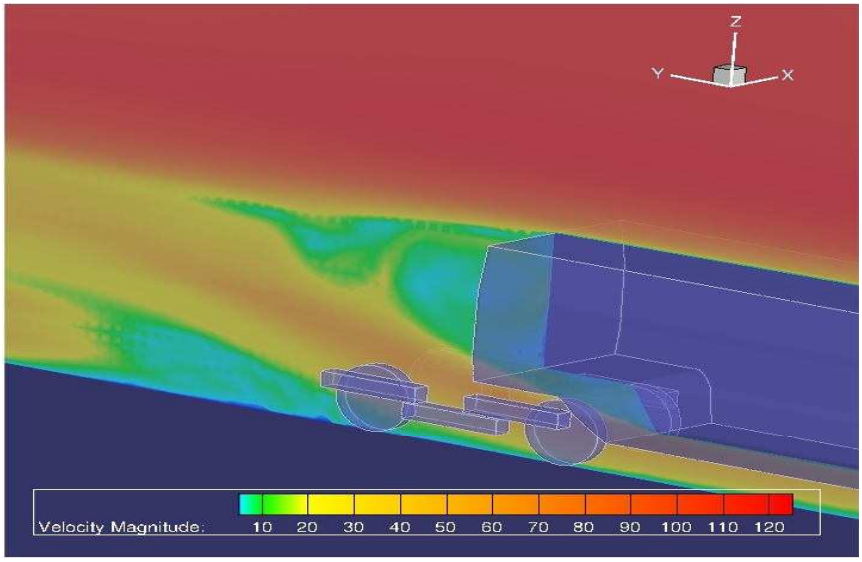

Fig. 6 The velocity distribution at the end of the train at $t = 1.0s$. Here $v_{in} = 100m/s$, $\mu = 1.83 \times 10^{-5} pa \cdot s$, $Re = 10^7$.

Table 4 Comparison of different local solver for the high speed train simulation. Here $DOF = 1.7 \times 10^7$ and $Re = 10^7$. The fill-in level of the ILU is 4.

n_p	Newton		GMRES		Time	
	LU	ILU(4)	LU	ILU(4)	LU	ILU(4)
1024	4.0	4.0	55.3	73.6	1501.2	395.2
1536	4.0	4.0	59.6	79.8	823.0	327.1
2048	4.0	4.0	62.1	82.0	420.6	239.5
3072	4.0	4.0	66.3	89.5	273.1	182.7

As we mentioned in Section 4, an alternative approach is point-block ILU. We show a comparison of the different subdomain solvers in Table 4 for this test case, and this table reveals that while ILU takes more GMRES iterations than LU, it takes much less compute time than LU, especially when the number of processors is small.

For the overlapping Schwarz preconditioner, an important parameter that influences the strength of the preconditioner is the overlapping size δ. From the theory of the overlapping Schwarz method, larger overlap often implies a faster convergence (fewer GMRES iterations), at least for elliptic systems with sufficient regularity [28]. However, larger overlap also means larger subdomain problems and more information transfer between subdomains, as a result, the overall com-

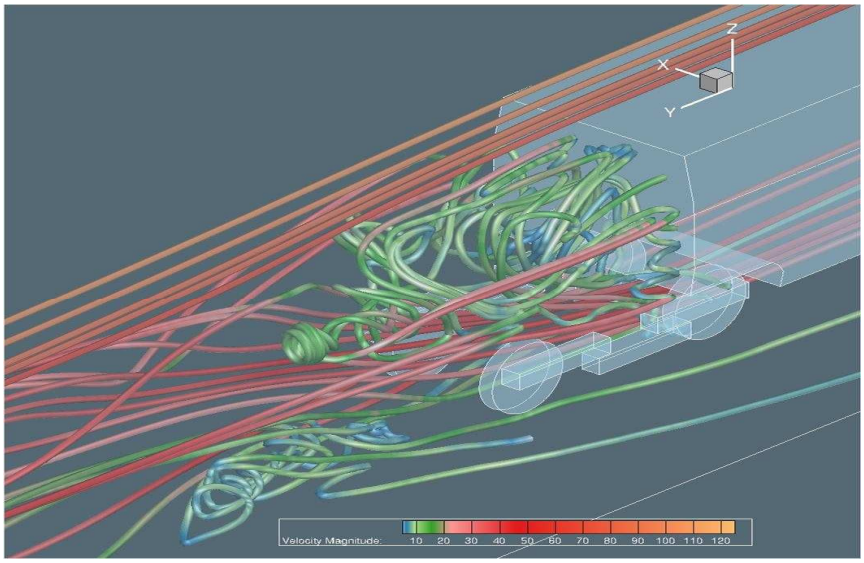

Fig. 7 The stream trace of the flow at the end of the train at $t = 1.0s$. Here $v_{in} = 100m/s$, $\mu = 1.83 \times 10^{-5} pa \cdot s$, $Re = 10^7$.

Table 5 The effect of various choices of the overlapping parameter δ for the high speed train simulation. Here $DOF = 8 \times 10^6$ and $Re = 10^7$.

δ	n_p	Newton	GMRES	Time
2	1024	4.0	209.6	338.0
4	1024	4.0	131.3	410.1
6	1024	4.0	91.4	510.6
2	2048	4.0	301.5	133.8
4	2048	4.0	169.2	175.7
6	2048	4.0	136.2	262.7

pute time may increase. Table 5 shows the effect of the overlapping parameter for solving the high speed train simulation problem. The best result is obtained with $\delta = 2$. For $\delta = 0,\ 1$, the preconditioner is so weak and the algorithm doesn't converge for some cases.

Besides the parallel performance and scalability, the robustness with respect to the Reynolds number Re is another important consideration in the design of solution algorithms for the flow simulation problems. Table 6 shows that the algorithm is quite robust for a wide range of Reynolds numbers.

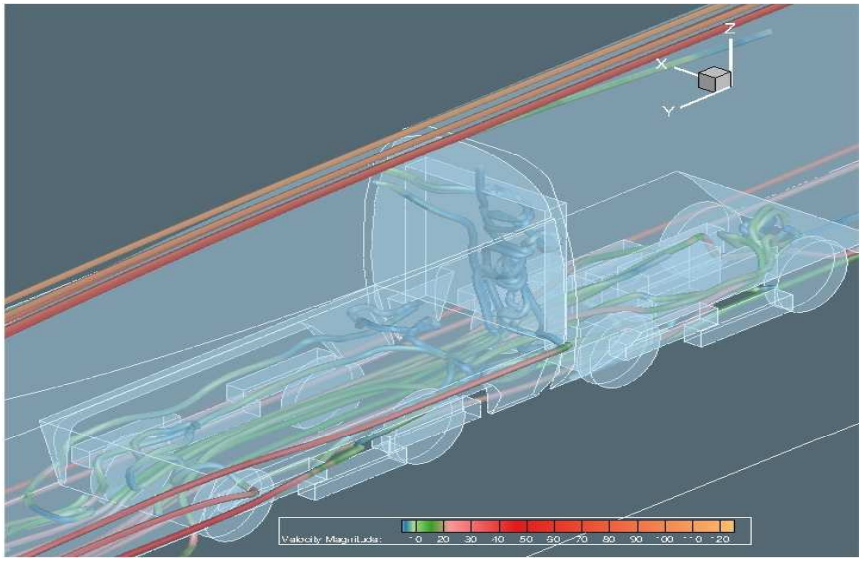

Fig. 8 The stream trace of the flow around the wheels of the train at $t = 1.0s$. Here $v_{in} = 100m/s$, $\mu = 1.83 \times 10^{-5} pa \cdot s$, $Re = 10^7$.

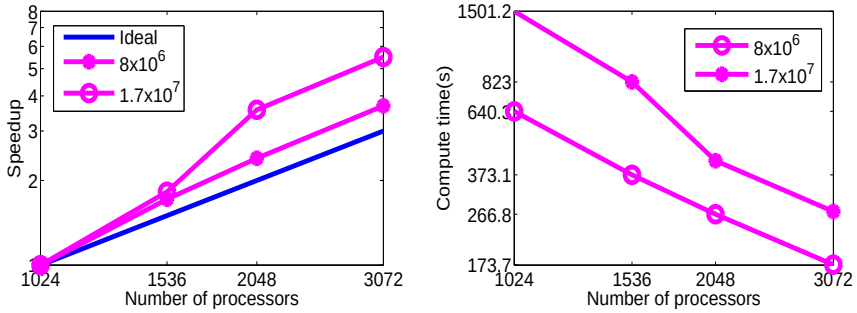

Fig. 9 The speedup and the average compute time per time step (log-log scaled) for the high speed train simulation. Here $DOF = 8 \times 10^6$ and $Re = 10^7$.

6 Concluding remarks

A domain decomposition based parallel algorithm for the direct numerical simulation of complex flows is introduced and studied in this paper. The algorithm begins with a fully implicit discretization of the unsteady incompressible Navier-Stokes equations on an unstructured mesh with a stabilized finite element method, then

Table 6 The robustness of the algorithm with respect to the Reynolds number Re for the train simulation. Here $DOF = 8 \times 10^6$.

Re	n_p	Newton	GMRES	Time
1.0×10^5	1024	3.0	109.1	390.4
1.0×10^6	1024	3.0	99.5	385.5
1.0×10^7	1024	4.0	91.4	512.7
1.0×10^5	2048	3.0	137.7	195.4
1.0×10^6	2048	3.0	129.9	195.1
1.0×10^7	2048	4.0	136.2	261.1

an inexact Newton method is employed to solve the large nonlinear system at each time step, and a preconditioned GMRES method is employed to solve the linear Jacobian system in each Newton step with a one-level additive Schwarz preconditioner. We tested the algorithm for a benchmark problem and a high speed train simulation problem with more than 17 million degrees of freedom. The numerical experiments showed that the method has a superlinear speedup with up to 3072 processors.

References

1. Alfonsi, G.: Reynolds-averaged Navier-Stokes equations for turbulence modeling. Appl. Mech. Rev. 62, 040802 (2009)
2. Alfonsi, G.: On direct numerical simulation of turbulent flows. Appl. Mech. Rev. 64, 020802 (2011)
3. Balay, S., Buschelman, K., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Smith, B.F., Zhang, H.: PETSc Users Manual. Tech. Rep., Argonne National Laboratory (2012)
4. Bayraktar, E., Mierka, O., Turek, S.: Benchmark computations of 3D laminar flow around a cylinder with CFX, OpenFOAM and FeatFlow. Int. J. Comput. Sci. Engrg. 7, 253-266 (2012)
5. Bazilevs, Y., Calo, V.M., Hughes, T.J.R., Zhang, Y.: Isogeometric fluid-structure interaction: theory, algorithms, and computations. Comput. Mech. 43, 3-37 (2008)
6. Cai, X.-C., Gropp, W.D., Keyes, D.E., Melvin, R.G., Young, D.P.: Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. SIAM J. Sci. Comput. 19, 246-265 (1998)
7. Cai, X.-C., Sarkis, M.: A restricted additive Schwarz preconditioner for general sparse linear systems. SIAM J. Sci. Comput. 21, 792-797 (1999)
8. Chan, T.F., van der Vorst, H.A.: Approximate and incomplete factorizations. In Parallel Numerical Algorithms, ICASE/LaRC Interdisciplinary Series in Science and Engineering IV. Centenary Conference, Keyes, D.E., Sameh, A., Venkatakrishnan, V. eds. Dordrecht. Kluver Academic Publishers, 167-202 (1997)
9. Chen, J.H.: Petascale direct numerical simulation of turbulent combustion-fundamental insights towards predictive models. Proc. Combust. Inst. 33, 99-123 (2011)
10. Eisenstat, S.C., Walker, H.F.: Choosing the forcing terms in an inexact Newton method. SIAM J. Sci. Comput. 17, 16-32 (1996)
11. Franca, L.P., Frey, S.L.: Stabilized finite element method: II. The incompressible Navier-Stokes equation. Comput. Methods Appl. Mech. Engrg. 99, 209-233 (1992)
12. Friedrich, R., Huttl, T.J., Manhart, M., Wagner, C.: Direct numerical simulation of incompressible turbulent flows. Comput. Fluids 30, 555-579 (2001)
13. Guermond,J.-L., Oden, J.T., Prudhomme, S.: Mathematical perspectives on large eddy simulation models for turbulent flows. J. Math. Fluid Mech. 6, 194-248 (2004)
14. Hwang, F.-N., Cai, X.-C.: A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations. J. Comput. Phys. 204, 666-691 (2005)

15. Hwang, F.-N., Wu, C.-Y., Cai, X.-C.: Numerical simulation of three-dimensional blood flows using domain decomposition method on parallel computer. J. Chinese. Soc. Mech. Engrg. 31, 199-208 (2010)
16. John, V.: On the efficiency of linearization schemes and coupled multigrid methods in the simulation of a 3D flow around a cylinder. Int. J. Numer. Meth. Fluids 50, 845-862 (2006)
17. Karypis, G.: METIS/ParMETIS webpage. University of Minnesota, `http://glaros.dtc.umn.edu/gkhome/views/metis` (2012)
18. Mahesh, K., Costantinescu, G., Moin, P.: A numerical method for large-eddy simulation in complex geometries. J. Comput. Phys. 197, 215-240 (2004)
19. Moin, P., Mahesh, K.: Direct numerical simulation: a tool in turbulence research. Annu. Rev. Fluid Mech. 30, 539-578 (1998)
20. Murillo, M., Cai, X.-C.: A fully implicit parallel algorithm for simulating the nonlinear electrical activity of the heart. Numer. Linear Algebra. Appl. 11, 261-277 (2004)
21. Owen, S.J., Shepherd, J.F.: CUBIT project webpage. `http://cubit.sandia.gov/` (2012)
22. Piomelli, U.: Large-eddy simulation: achievements and challenges. Prog. Aeosp. Sci., 35, 335-362 (1999)
23. Rahimian, A., Lashuk, I., Veerapaneni, S., Chandramowlishwaran, A., Malhotra, D., Moon, L., Sampath, R., Shringarpure, A., Vetter, J., Vuduc, R., Zorin D., Biros, G.: Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. Proc. ACM/IEEE Supercomput. Conf., 2010
24. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing Company, Boston (1996)
25. Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. 7, 856-869 (1986)
26. Sagaut, P.: Large eddy simulation for incompressible flows. Springer-Verlag, Berlin (2000)
27. Schäfer, M., Turek, S.: Benchmark computations of laminar flow around a cylinder. Notes Numer. Fluid Mech. 52, 547-566 (1996)
28. Toselli, A., Widlund, O.: Domain Decomposition Methods: Algorithms and Theory. Springer-Verlag, Berlin (2005)
29. Yokokawa, M., Itakura, K.I., Uno, A., Ishihara, T., Kaneda, Y.: 16.4-Tflops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator. Proc. ACM/IEEE Supercomput. Conf., 2002