INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **53**:947–968
Published online 23 August 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1314

A methodology for high performance computation of fully inhomogeneous turbulent flows

Donghyun You[1,*,†], Meng Wang[2] and Rajat Mittal[3]

[1]*Center for Turbulence Research, Stanford University, Stanford, California 94305, U.S.A.*
[2]*Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, Indiana 46556, U.S.A.*
[3]*Department of Mechanical and Aerospace Engineering, George Washington University, Washington, District of Columbia 20052, U.S.A.*

SUMMARY

A large-eddy simulation methodology for high performance parallel computation of statistically fully inhomogeneous turbulent flows on structured grids is presented. Strategies and algorithms to improve the memory efficiency as well as the parallel performance of the subgrid-scale model, the factored scheme, and the Poisson solver on shared-memory parallel platforms are proposed and evaluated. A novel combination of one-dimensional red–black/line Gauss–Seidel and two-dimensional red–black/line Gauss–Seidel methods is shown to provide high efficiency and performance for multigrid relaxation of the Poisson equation. Parallel speedups are measured on various shared-distributed memory systems. Validations of the code are performed in large-eddy simulations of turbulent flows through a straight channel and a square duct. Results obtained from the present solver employing a Lagrangian dynamic subgrid-scale model show good agreements with other available data. The capability of the code for more complex flows is assessed by performing a large-eddy simulation of the tip-leakage flow in a linear cascade. Copyright © 2006 John Wiley & Sons, Ltd.

Received 15 November 2005; Revised 15 June 2006; Accepted 22 June 2006

KEY WORDS: large-eddy simulation; Navier–Stokes equations; inhomogeneous turbulent flow; multigrid method; high performance computation; OpenMP

1. INTRODUCTION

With increasing computing power, applicability of computational fluid dynamics (CFD) has extended to more realistic and complex problems from traditional block body problems. In particular,

*Correspondence to: D. You, Center for Turbulence Research, Stanford University, Stanford, California 94305, U.S.A.
†E-mail: dyou@stanford.edu

Contract/grant sponsor: Office of Naval Research; contract/grant number: N00014-99-1-0389

the large-eddy simulation (LES) technique, in which resolved and subgrid-scale (SGS) motions are defined by a spatial filter applied to the Navier–Stokes equations, has emerged as a promising tool for time accurate simulations of turbulent flows. In most complex flow configurations of interest, the need for grid resolution and corresponding computational resources make their analysis often inadequate, except on large-scale parallel computers.

In addition to the need for computational resources, fully inhomogeneous complex turbulent flows necessitate a localized version of the SGS model [1–4], complicated boundary conditions [5, 6], and efficient computational algorithms. Especially when compared to configurations with at least one homogeneous flow direction, the full inhomogeneity prevents the use of fast Fourier transforms (FFT) and significantly increases computational overhead in the Poisson equation required for mass conservation of incompressible flow. Since the direct solution method for the Poisson equation is too expensive, the method has usually been substituted by an iterative scheme. The multigrid method has been widely adopted for this purpose due to its rapid convergence. However, its convergence property and parallelizability are significantly affected by the choice of base relaxation schemes and its optimization usually requires extensive numerical experiments.

During the course of developing the present code, it was found that the solution procedure for the Poisson equation is the most costly part in the entire computation and accounts for more than 60% of the total computational cost. Therefore, in this study, the design and optimization of the algorithm for the Poisson equation is considered with the highest priority. Although numerous relaxation algorithms for multigrid methods and their two-dimensional applications have been reported in the literature, much remains to be investigated regarding the extension of multigrid algorithms to three-dimensional space and their evaluation, especially when they are coupled with parallelization issues. The present study also intends to implement a more efficient three-dimensional multigrid solver by experimenting with relaxation schemes in common use such as red–black Gauss–Seidel, line Gauss–Seidel methods, and their combinations, with which the memory efficiency, parallel performance, and favourable convergence property are satisfied.

Parallel performance is another important factor in the code development. Among the various parallelization methodologies, message passing interface (MPI) (e.g. References [7–9]) and multi-processing directives (OpenMP) (e.g. References [10–13]) are currently the dominant tools for parallelization. MPI implementation has an advantage in the portability of the code regardless of the computer platform's specific memory architecture. However, it often requires much effort in writing a new code or porting a serial code to a parallel one. Distributing data structure to CPUs is the responsibility of the programmers, who must find efficient ways to accomplish the needed data shuffling [14].

Recently, with the introduction of large-scale shared or shared-distributed combined memory systems, OpenMP has emerged as an alternative to MPI on shared memory platforms. Since OpenMP allows a serial code to be parallelized by inserting predefined parallelization directives to the serial code, usually the implementation is easier than MPI parallelization. Also, data distribution issues never arise in the shared memory implementation. A programmer needs only to make sure that the CPUs have an equal amount of work to do with efficient access of the shared memory [15, 16].

However, the parallel performance of OpenMP implementation observed in the literature shows rapid saturation as the number of CPUs increases [10, 17]. Except for system based issues such as nonuniform memory access (NUMA) and the finite bandwidth of a common bus of certain systems, the lack of strategies for shared-memory parallelization has been usually identified as a main cause for the parallel performance degradation [10, 11].

DOI: 10.1002/fld

The present LES solver adopts OpenMP as a parallelization tool to accommodate the shared and shared-distributed systems provided by the U.S. Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP). Since an efficient memory utilization is the key factor for parallel speedup in the OpenMP parallelization, designing an algorithm which retains memory efficiency is essential to a LES of large-scale complex flows. Therefore, the main effort is devoted to the optimal data distribution among the CPUs and memory management for large-scale applications.

The main objectives of this study are: (i) to develop highly efficient algorithms including the multigrid method for the Poisson solver; (ii) to optimize the algorithms for high parallel performance; and (iii) to evaluate the feasibility of the newly developed code for LES of complex, statistically fully inhomogeneous turbulent flows.

In what follows, details of the computational aspects of the LES solver are described. This is followed by a discussion on strategies for parallelizing the main parts of the solver, and discussions on benchmark simulations performed to evaluate the performance and capability of the newly developed solver.

2. COMPUTATIONAL METHOD

2.1. Numerical method

The spatially filtered Navier–Stokes equations for resolved scales in LES are as follows:

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\overline{u}_i\overline{u}_j = -\frac{\partial \overline{p}}{\partial x_i} + \frac{1}{Re}\frac{\partial}{\partial x_j}\frac{\partial \overline{u}_i}{\partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} \tag{1}$$

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \tag{2}$$

where τ_{ij} is the SGS stress tensor and $\overline{()}$ denotes a filtered quantity. All the coordinate variables x_i, velocity components u_i, and pressure p are non-dimensionalized by the length scale L, the reference velocity U_{ref}, and ρU_{ref}^2, respectively. The time is normalized by L/U_{ref}. Re denotes the Reynolds number and is defined as $Re = \rho U_{\text{ref}} L/\mu$ where μ is molecular viscosity.

The key feature of the numerical method is the use of a nondissipative, central-difference spatial discretization scheme which has been demonstrated to be crucial for maintaining the stability and accuracy of the LES approach over a range of spatial and temporal scales [18, 19]. In our LES approach, aliasing errors are controlled by enforcing kinetic energy conservation, not by numerical dissipation or filtering, and artificial damping of small scales is avoided. It has been reported that numerical dissipation due to the use of upwind-biased schemes often overwhelms the SGS model terms, and that the nondissipative second-order scheme even predicts better results than those obtained using higher-order upwind schemes at similar grid resolution [18, 19].

The SGS stress tensor τ_{ij} is modelled by a Smagorinsky type eddy-viscosity model:

$$\tau_{ij} - \tfrac{1}{3}\delta_{ij}\tau_{ij} = -2c\Delta^2|\overline{S}|\overline{S}_{ij} = -2\nu_t\overline{S}_{ij} \tag{3}$$

where δ_{ij}, Δ, S_{ij}, and ν_t are Kronecker delta, grid spacing, strain rate tensor, and eddy viscosity, respectively. To compute the Smagorinsky coefficient c, a Lagrangian dynamic model is employed [1]. Since the Lagrangian dynamic model averages the model coefficient along the flow pathlines

DOI: 10.1002/fld

as opposed to the homogeneous flow direction in the standard dynamic model [20], it is suitable for fully inhomogeneous complex flows. The equation for computing the coefficient is

$$c^2(\mathbf{x}, t) = \frac{\Phi_{LM}}{\Phi_{MM}} \tag{4}$$

where

$$\Phi_{LM} = \frac{1}{T} \int_{-\infty}^{t} L_{ij} M_{ij}(\mathbf{z}(t), t') \mathrm{e}^{-(t-t')/T} \, \mathrm{d}t' \tag{5}$$

$$\Phi_{MM} = \frac{1}{T} \int_{-\infty}^{t} M_{ij} M_{ij}(\mathbf{z}(t), t') \mathrm{e}^{-(t-t')/T} \, \mathrm{d}t' \tag{6}$$

The exponential weighting function $(1/T)\mathrm{e}^{-(t-t')/T}$ is introduced in order to average $L_{ij}M_{ij}$ and $M_{ij}M_{ij}$ over pathlines with an exponentially decreasing weight in backward time. T is an averaging time scale and is expressed as

$$T = \theta \Delta (\Phi_{LM} \Phi_{MM})^{-1/8} \tag{7}$$

where θ is a free parameter to be determined empirically.

L_{ij}, M_{ij}, and $\mathbf{z}$ are defined as follows:

$$L_{ij} = \widehat{\overline{u}_i \overline{u}_j} - \widehat{\overline{u}}_i \widehat{\overline{u}}_j \tag{8}$$

$$M_{ij} = 2\Delta^2 (\widehat{|\overline{S}|\overline{S}}_{ij} - 4|\widehat{\overline{S}}|\widehat{\overline{S}}_{ij}) \tag{9}$$

$$\mathbf{z}(t') = \mathbf{x}(t) - \int_{t'}^{t} \overline{\mathbf{u}}(\mathbf{z}(t''), t'') \, \mathrm{d}t'' \tag{10}$$

where $\widehat{()}$ denotes a test-filtered quantity.

The time-integration method used to solve the transformed governing equations is based on a fully implicit fractional-step method employing the Crank–Nicolson scheme. The coordinate transformation and discretization of the governing equations are described in detail in the following sections.

2.2. Momentum equations

The Navier–Stokes equations can be expressed in generalized coordinate as

$$\frac{\partial q^i}{\partial t} = -N^i(\mathbf{q}) - G^i(p) + L^i(\mathbf{q}) \tag{11}$$

$$D^i q^i = 0 \tag{12}$$

where $\mathbf{q} = (q^1, q^2, q^3)$, N^i is the nonlinear convection term, $G^i(p)$ is the pressure gradient term, L^i is the diffusion term, and D^i is the divergence operator [21]. Generalized coordinates for the streamwise and wall-normal directions and a Cartesian coordinate for the spanwise nonuniform grid distribution are introduced in Figure 1 as

$$(x_1, x_2, x_3; u_1, u_2, u_3) \to (\eta^1, \eta^2, \eta^3; q^1, q^2, q^3) \tag{13}$$

DOI: 10.1002/fld

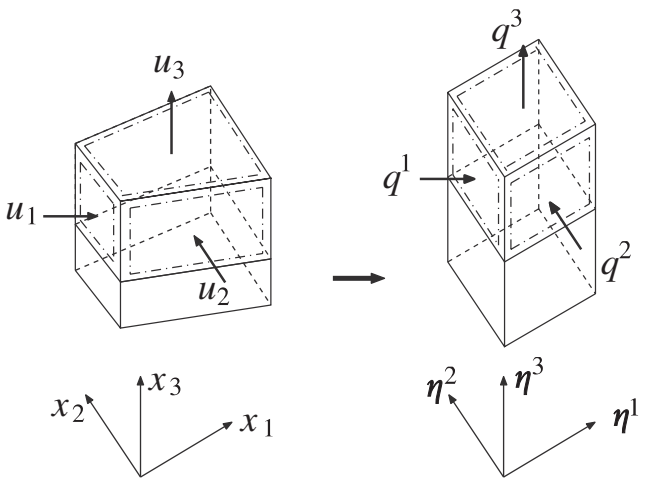

Figure 1. Schematic diagram of coordinate transformation from Cartesian coordinates to curvilinear coordinates.

The q^i is the volume flux across the faces of the cells, which is equivalent to the contravariant velocity components on a staggered grid multiplied by the Jacobian of the coordinate transformation. Then, each term in Equation (11) is expressed in the generalized coordinate as

$$N^i(\mathbf{q}) = \frac{1}{J}\gamma_m^i \frac{\partial}{\partial \eta^j}\frac{1}{J}c_k^m q^k q^j \tag{14}$$

$$G^i(p) = \alpha^{ij}\frac{\partial p}{\partial \eta^j} \tag{15}$$

$$L^i(\mathbf{q}) = \frac{1}{J}\gamma_m^i \frac{\partial}{\partial \eta^k}\alpha^{kj}\frac{1}{Re}\frac{\partial}{\partial \eta^j}\frac{1}{J}c_l^m q^l \tag{16}$$

where $q^j = \gamma_k^j u_k$, $c_k^j = \partial x_j/\partial \eta^k$, $\gamma_k^j = J(c_j^k)^{-1}$, $\alpha^{jk} = J(c_j^m c_k^m)^{-1}$, and $J = \det(\partial x_j/\partial \eta^k)$ for $j, k, l, m = 1, 2, 3$.

Constraining the third (η^3) direction as a Cartesian coordinate leads to $c_3^1 = c_3^2 = \gamma_3^1 = \gamma_3^2 = 0$ and $J = (c_1^1 c_2^2 - c_2^1 c_1^2)c_3^3$, and necessitates only a set of two-dimensional arrays for c_1^1, c_1^2, c_2^1, and c_2^2 and a one-dimensional array for c_3^3.

All terms including the cross-derivative diffusion terms are advanced in time using the Crank–Nicolson method and are discretized by second-order central differences in space. The fully-implicit, fractional-step method applied to Equations (11) and (12) results in

$$\frac{\hat{q^i} - q^{i^n}}{\Delta t} = \frac{1}{2}(L^i(\hat{\mathbf{q}}) + L^i(\mathbf{q}^n)) - \frac{1}{2}(N^i(\hat{\mathbf{q}}) + N^i(\mathbf{q}^n)) \tag{17}$$

$$\frac{q^{i^{n+1}} - \hat{q^i}}{\Delta t} = -G^i(\phi^{n+1}) \tag{18}$$

DOI: 10.1002/fld

with

$$D^i q^{i^{n+1}} = 0 \tag{19}$$

where ϕ is a scalar to be determined. Equation (17) is a second-order-accurate fully-implicit treatment of Equation (11) with $G^i(p)$ excluded. Substituting Equation (18) into Equation (17) shows that the overall accuracy is still second order. Here, ϕ is referred to as the pseudo-pressure and, in fact, is different from the original pressure by

$$G^i(p^{n+1}) = G^i(\phi^{n+1}) - \tfrac{1}{2}\Delta t L^i(\mathbf{G}(\phi^{n+1})) + \tfrac{1}{2}[N^i(\mathbf{q}^{n+1} + \Delta t \mathbf{G}(\phi^{n+1})) - N^i(\mathbf{q}^{n+1})] \tag{20}$$

where $\mathbf{G} = (G^1, G^2, G^3)$. The detailed procedure to obtain ϕ^{n+1} is described in Section 2.3.

The discretized nonlinear equations are solved using a Newton-iterative method. The Newton-iterative method retains the nonlinearity of the equations and a quadratic convergence when the initial condition is near the solution. The intermediate flux, $\hat{q^i}$, is obtained by solving Equation (17), which is rewritten as

$$\hat{q^i} + \tfrac{1}{2}\Delta t(N^i(\hat{\mathbf{q}}) - L^i(\hat{\mathbf{q}})) = q^{i^n} - \tfrac{1}{2}\Delta t(N^i(\mathbf{q}^n) - L^i(\mathbf{q}^n)) \equiv R^{i^n} \tag{21}$$

or

$$F^i(\hat{\mathbf{q}}) = \hat{q^i} + \tfrac{1}{2}\Delta t(N^i(\hat{\mathbf{q}}) - L^i(\hat{\mathbf{q}})) - R^{i^n} = 0 \tag{22}$$

Applying the Newton-iterative method to Equation (22) gives

$$\left\{\frac{\partial F^i(\hat{\mathbf{q}})}{\partial \hat{q^j}}\right\}^r \delta \hat{q^j}^{r+1} = -F^i(\hat{\mathbf{q}}^r) \tag{23}$$

where $\delta \hat{q^j}^{r+1} = \hat{q^j}^{r+1} - \hat{q^j}^r$, r is the iteration index, and $j = 1, 2, 3$.

Equation (23) becomes

$$\left\{\delta_{ij} + \frac{\partial}{\partial \hat{q^j}}\left(\frac{1}{2}\Delta t(N^i(\hat{\mathbf{q}}) - L^i(\hat{\mathbf{q}}))\right)\right\}^r \delta \hat{q^j}^{r+1} = -F^i(\hat{\mathbf{q}}^r) \tag{24}$$

and we introduce matrices

$$M_{ij} = \left\{\frac{\partial}{\partial \hat{q^j}}(N^i(\hat{\mathbf{q}}) - L_1^i(\hat{\mathbf{q}}))\right\} \tag{25}$$

$$Q_{ij} = \left\{\frac{\partial}{\partial \hat{q^j}}(-L_2^i(\hat{\mathbf{q}}))\right\} \tag{26}$$

where L_1^i and L_2^i are the diffusion terms without and with cross derivatives, respectively. Now we split $M_{ij} (= M_{ij}^1 + M_{ij}^2 + M_{ij}^3)$ into three parts, each containing η^1-, η^2-, and η^3-derivatives,

DOI: 10.1002/fld

respectively. Using an approximate factorization technique, Equation (24) becomes

$$\left(1+\frac{1}{2}\Delta t M^1_{\alpha\alpha}\right)^r \left(1+\frac{1}{2}\Delta t M^2_{\alpha\alpha}\right)^r \left(1+\frac{1}{2}\Delta t M^3_{\alpha\alpha}\right)^r \delta\hat{q}^{\alpha r+1}$$

$$= -F^{\alpha}(\hat{\mathbf{q}}^r) - \frac{1}{2}\kappa\Delta t \sum_{l=1}^{3} (M^l_{\alpha j})^r \delta\hat{q}^{j*} - \frac{1}{2}\Delta t (Q_{\alpha j})^r \delta\hat{q}^{j*} \tag{27}$$

where $\kappa = 1$ for $j \neq \alpha$ and $\kappa = 0$ for $j = \alpha$, with $j = 1, 2, 3$. No summation is implied in repeated α (α = 1, 2, or 3). $\delta\hat{q}^{j*}$ is updated during the iteration step. This requires inversions of tridiagonal matrices which result in a significant reduction in computing cost and memory.

An inflow/outflow or periodic boundary condition is utilizable in the x_1 direction while periodic or Dirichlet boundary conditions can be applied in both the x_2 and x_3 directions.

2.3. Poisson equation

Equations (18) and (19) are combined to eliminate $q^{i^{n+1}}$, which results in a Poisson equation for ϕ^{n+1}:

$$D^i G^i(\phi^{n+1}) = \frac{1}{\Delta t} D^i \hat{q}^i \tag{28}$$

Since the Poisson solver is the most costly part in the computation of fully three-dimensional inhomogeneous flows, it is crucial to improve both the convergence and parallel efficiency for large-scale computations. Multigrid methods have been popular for this purpose and have shown their competence in various applications [22]. However, using multigrids in all three dimensions requires several times larger memory than the amount needed for a base grid and complicates the coarsening procedure. On the contrary, two-dimensional multigrid coarsening (semi-coarsening) with a direct solution procedure in the remaining direction reduces the memory requirement considerably while its optimization is much easier. Both the line Gauss–Seidel and red–black Gauss–Seidel methods have attractive features for the latter strategy and they have also shown good convergence properties and parallelizability in a number of various applications [23–27]. The advantage of semi-coarsening strategy is observed to be more significant as the flow anisotropy becomes more prominent. For flow of which characteristics is mainly isotropic and homogeneous, a fully three-dimensional coarsening or a Krylov based algorithm can be considered as an alternative to the semi-coarsening strategy.

Equation (28) can be discretized as

$$\mathscr{L}_h \phi^{n+1}_{i,j,k} = \hat{f}_h \tag{29}$$

where $\mathscr{L}_h$ is the second-order central-difference operator for $D^i G^i$, $\hat{f}_h = (1/\Delta t) D^i \hat{q}^i$, and h denotes the grid spacing. For an approximate solution $\widetilde{\phi}_{i,j,k}$ to Equation (29), the correction (φ_h) and the residual (r_h) are defined as

$$\varphi_h = \phi^{n+1}_{i,j,k} - \widetilde{\phi}_{i,j,k} \tag{30}$$

$$r_h = \mathscr{L}_h \widetilde{\phi}_{i,j,k} - \hat{f}_h \tag{31}$$

DOI: 10.1002/fld

Since the $\mathscr{L}_h$ is a linear operator, Equations (30) and (31) satisfy

$$\mathscr{L}_h \varphi_h = -r_h \tag{32}$$

In the multigrid algorithm, the residual equation (32) is approximated by

$$\mathscr{L}_H \varphi_H = -r_H \tag{33}$$

where $\mathscr{L}_H$ and φ_H are the discrete Laplace operator and solution on a coarser grid. To define the residual r_H on the coarse grid, a restriction operator $\mathscr{R}$ is employed such that

$$r_H = \mathscr{R} r_h \tag{34}$$

Once a solution $\widetilde{\varphi}_H$ to Equation (33) is obtained, a prolongation operator $\mathscr{P}$ is applied to interpolate the correction to the fine grid as

$$\widetilde{\varphi}_h = \mathscr{P} \widetilde{\varphi}_H \tag{35}$$

Finally, the approximation $\widetilde{\phi}_{i,j,k}$ is updated as

$$\widetilde{\phi}^{\text{new}}_{i,j,k} = \widetilde{\phi}_{i,j,k} + \widetilde{\varphi}_h \tag{36}$$

A bilinear interpolation is employed for the prolongation operation ($\mathscr{P}$) while the adjoint operator to $\mathscr{P}$ is used for restriction ($\mathscr{R}$) (see Reference [28] for details of the operators). The steps in Equations (30)–(36) are repeated during a V- or W-cycle [28] multigrid iterations until a required convergence is achieved.

In this study, two possible combinations of the line and red–black Gauss–Seidel methods are considered as the smoothing operator for Equations (32) and (33). The first approach (1DRBL) consists of a one-dimensional red–black method in the η^1–η^3 (i–k) plane and a line Gauss–Seidel method in the η^1–η^2 (i–j) plane (see Figure 2). The line Gauss–Seidel approach in the i–j plane results in an even–odd decomposition in i indices, and then each ith vector is solved using a tridiagonal matrix solver in the j-direction. It is worth noting that in the present coordinate system, partitioning the computational domain along the k-direction is beneficial in terms of load balancing since the grids in x–y planes are not altered along the k-direction. The detailed steps at a certain level of the multigrid cycle on a uniform structured mesh of grid spacing H are summarized as

Step 1: $k = \text{odd}$ and $i = \text{odd}$

$$\widetilde{\varphi}^{s1}_{j-H,i,k} - 6\widetilde{\varphi}^{s1}_{j,i,k} + \widetilde{\varphi}^{s1}_{j+H,i,k} = -\varphi_{j,i-H,k} - \varphi_{j,i+H,k} - \varphi_{j,i,k-H} - \varphi_{j,i,k+H} - H^2 r_{i,j,k}$$

Step 2: $k = \text{odd}$ and $i = \text{even}$

$$\widetilde{\varphi}^{s2}_{j-H,i,k} - 6\widetilde{\varphi}^{s2}_{j,i,k} + \widetilde{\varphi}^{s2}_{j+H,i,k} = -\widetilde{\varphi}^{s1}_{j,i-H,k} - \widetilde{\varphi}^{s1}_{j,i+H,k} - \varphi_{j,i,k-H} - \varphi_{j,i,k+H} - H^2 r_{i,j,k}$$

Step 3: $k = \text{even}$ and $i = \text{odd}$

$$\widetilde{\varphi}^{s3}_{j-H,i,k} - 6\widetilde{\varphi}^{s3}_{j,i,k} + \widetilde{\varphi}^{s3}_{j+H,i,k} = -\varphi_{j,i-H,k} - \varphi_{j,i+H,k} - \widetilde{\varphi}^{s1}_{j,i,k-H} - \widetilde{\varphi}^{s1}_{j,i,k+H} - H^2 r_{i,j,k}$$

Step 4: $k = \text{even}$ and $i = \text{even}$

$$\widetilde{\varphi}^{s4}_{j-H,i,k} - 6\widetilde{\varphi}^{s4}_{j,i,k} + \widetilde{\varphi}^{s4}_{j+H,i,k} = -\widetilde{\varphi}^{s3}_{j,i-H,k} - \widetilde{\varphi}^{s3}_{j,i+H,k} - \widetilde{\varphi}^{s2}_{j,i,k-H} - \widetilde{\varphi}^{s2}_{j,i,k+H} - H^2 r_{i,j,k}$$

The second approach (2DRBL) consists of a two-dimensional red–black ordering as shown in Figure 3(a). Then, the (i, k)th vector is solved using a tridiagonal matrix solver in the j-direction

DOI: 10.1002/fld

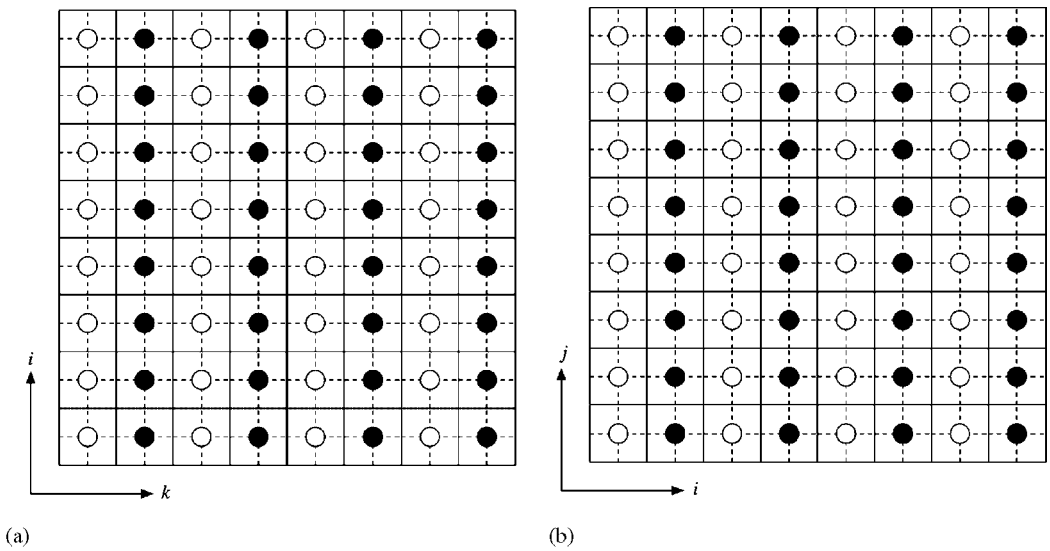

Figure 2. Schematic representation of the multigrid method (1DRBL) which consists of: (a) one-dimensional red–black Gauss–Seidel method in the i–k plane; and (b) line Gauss–Seidel method in the i–j plane.

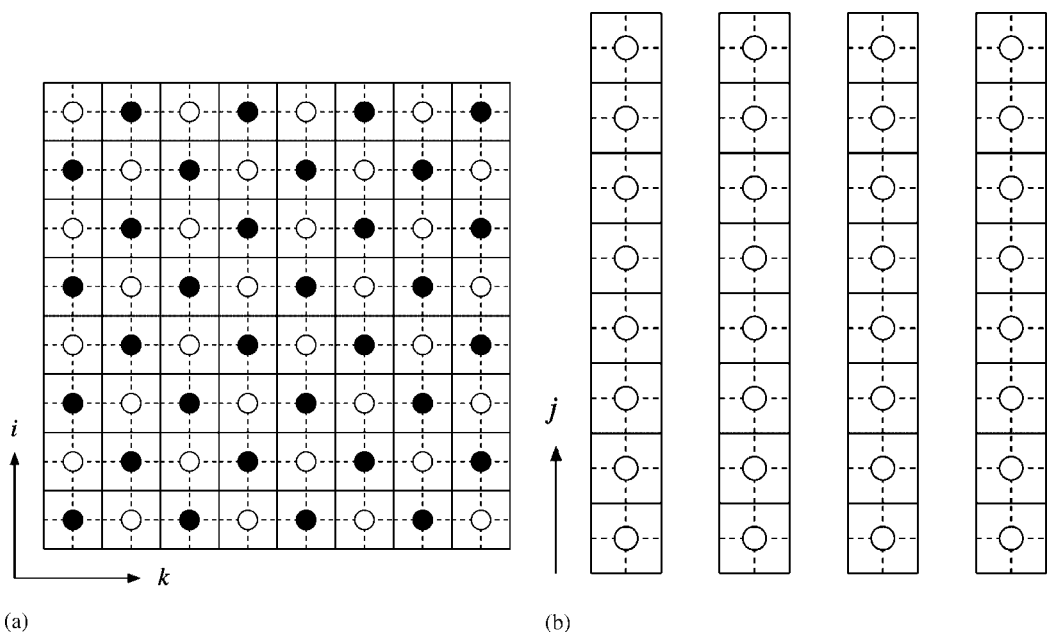

Figure 3. Schematic representation of the multigrid method (2DRBL) which consists of: (a) two-dimensional red–black Gauss–Seidel method in the i–k plane; and (b) tridiagonal matrix solver in the j-direction.

DOI: 10.1002/fld

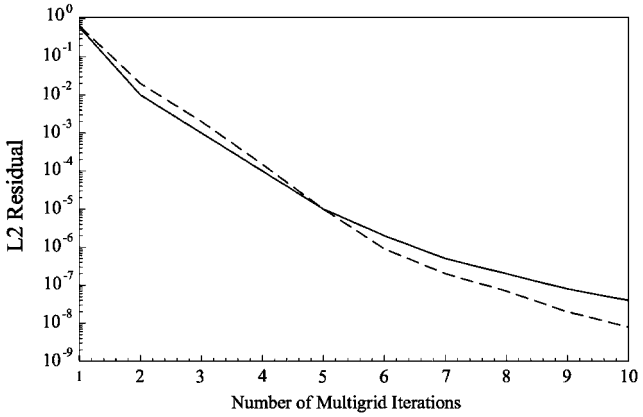

Figure 4. Convergence histories of proposed multigrid methods. —, 1DRBL; – – –, 2DRBL.

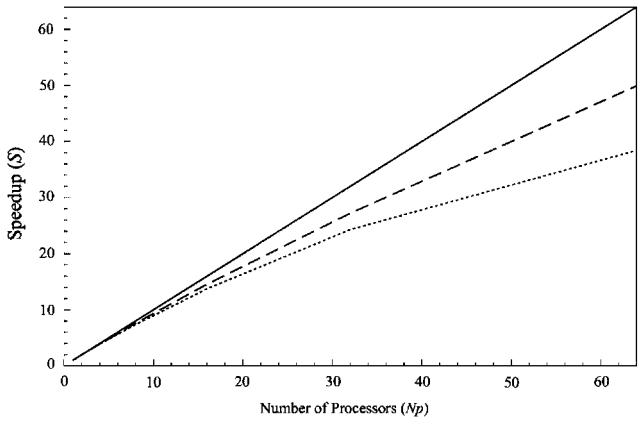

Figure 5. Parallel speedups of proposed multigrid methods on SGI Origin 3800. —, linear speedup; – – –, 1DRBL; · · ·, 2DRBL.

(Figure 3(b)). The two-step algorithm is

Step 1: red-colour

$$\tilde{\varphi}^{s1}_{j-H,i,k} - 6\tilde{\varphi}^{s1}_{j,i,k} + \tilde{\varphi}^{s1}_{j+H,i,k} = -\varphi_{j,i-H,k} - \varphi_{j,i+H,k} - \varphi_{j,i,k-H} - \varphi_{j,i,k+H} - H^2 r_{i,j,k}$$

Step 2: black-colour

$$\tilde{\varphi}^{s2}_{j-H,i,k} - 6\tilde{\varphi}^{s2}_{j,i,k} + \tilde{\varphi}^{s2}_{j+H,i,k} = -\tilde{\varphi}^{s1}_{j,i-H,k} - \tilde{\varphi}^{s1}_{j,i+H,k} - \tilde{\varphi}^{s1}_{j,i,k-H} - \tilde{\varphi}^{s1}_{j,i,k+H} - H^2 r_{i,j,k}$$

The same coarsening strategy and restriction/prolongation levels show similar convergence trends for both combinations (see Figure 4). Test simulations show that the second approach has a slightly better convergence property while the first approach shows a better parallel speedup (Figure 5). The utilization of these approaches for enhancing the parallel performance is discussed in Section 3.2.

DOI: 10.1002/fld

3. PARALLELIZATION AND OPTIMIZATION

3.1. Momentum equations

The computation of the momentum equations is initiated with the computation of eddy viscosity (v_t in Equation (3)). The dynamic procedure to obtain the eddy viscosity involves filtering operations of each velocity and strain rate tensor components, which in the standard shared memory approach, increases the stack memory requirement considerably (see Equations (3)–(10) for the detailed procedure).

To overcome this difficulty, as is often used in distributed memory parallelization, the computational domain is decomposed by the number of CPUs available. The resulting simplified algorithm looks like:

```
...
!$OMP PARALLEL PRIVATE (myid, mydomain, padding)
      myid = OMP_GET_THREAD_NUM()
      mydomain = totaldomain/OMP_GET_NUM_THREADS()
      mydomain = mydomain + padding
      CALL SGS(myid, mydomain)
!$OMP END PARALLEL
...
SUBROUTINE SGS(myid, mydomain)
SHARED(velocity, grid, metric-coefficients, eddy-viscosity)
SHARED(LM (Equation (5)), MM (Equation (6)))
...
    DO i, j, k = mydomain
      operations needed for Equations (3)–(10)
      eddy-viscosity = ...
    ENDDO
...
END SUBROUTINE SGS
```

Two or three mesh points are added as a padding to overlap data dependency regions for filtering as well as for the flow pathline tracking needed in the Lagrangian modelling.

Once a three-dimensional array containing the eddy viscosity is obtained, the Newton iteration to obtain an intermediate velocity field starts from the computation of the R^i terms in Equation (22). These terms are evaluated using flow-field quantities at the previous time step and all the computations are enclosed by do-loops with OpenMP directives in the outermost loop. To achieve improved flexibility and efficiency in memory handling, dynamic memory allocation/deallocation is utilized.

The factored equation given in Equation (27) consists of two main computational parts. First, the right-hand side term, $-F^i(\hat{\mathbf{q}}^r)$, in Equation (22) is computed in the same way as that used in the computation of R^i. Then, inversion of three tridiagonal matrices gives updated $\hat{q}^i$s. Considering that the inversions are performed along each of the η^1, η^2, and η^3 directions, loop-ordering and temporary arrays are configured to locate the tridiagonal inversions to the innermost loop.

DOI: 10.1002/fld

Table I. Wallclock times and speedups measured with up to 64 CPUs of SGI Origin 3800 by the multigrid solvers employing 1DRBL and 2DRBL, respectively.

	1DRBL		2DRBL	
No. of CPUs	Wallclock (s)	Speedup	Wallclock (s)	Speedup
1	8160	1.000	7991	1.000
2	4090	1.995	4007	1.994
4	2110	3.867	2070	3.860
8	1074	7.598	1086	7.358
16	560	14.571	581	13.754
32	301	27.110	329	24.289
64	163	50.061	209	38.234

The problem size is 257^3 and 3 multi-levels are used in both $x(i)$ and $z(k)$ directions for achieving a residual of 10^{-6}.

3.2. *Poisson equation*

In Section 2.3, three-dimensional multigrid methods which combine one- and two-dimensional red–black Gauss–Seidel and line Gauss–Seidel methods are constructed. In Figures 4 and 5, convergence properties and parallel performances of the multigrid solvers employing the two alternative combinations for solving Equation (28) are compared. The flow configuration, mesh size, and boundary conditions are described in Section 4.1 in detail. In this paper, the parallel speedup is defined as

$$\text{Speedup}(n) = \frac{T(1)}{T(n)} \tag{37}$$

where $T(n)$ represents wallclock time needed using n CPUs. Wallclock times have been measured using the OpenMP library function OMP_GET_WTIME(), and those taken by both methods to achieve the same residual defined as Equation (31) using up to 64 processors are compared in Table I. Test results in the present study represent the averaged values of 10 one-time step computations.

The 2DRBL shows a better result in wallclock time when the number of CPUs is small. However, as the number of CPUs exceeds 8, the combination of 1DRBL gives better results. Fast performance saturation due to reduced data locality and hence increased cache miss is observed in the 2DRBL while the maximum utilizable number of CPUs is limited by the number of lines at the highest multi-level in the 1DRBL. However, these disadvantages can be resolved by combining the two methods, which allows for a better parallel speedup at larger numbers of CPUs with fast convergence.

Considering that the 1DRBL is better than the 2DRBL in parallel performance, the 1DRBL is first applied up to the multi-level where all CPUs can be utilizable. Then the method switches to 2DRBL when the number of grid lines is smaller than the number of CPUs requested. This novel combination is also advantageous given the fact that the 2DRBL shows good convergence property with a coarser mesh. Note that the 'pressure' array and related loops of operations are constructed to locate the tridiagonal matrix inversions to the innermost level as done for the factored equation (see Section 2.2).

DOI: 10.1002/fld

4. CODE EVALUATION AND VALIDATION

4.1. Parallel performance

The parallel performance of the code has been evaluated in three different shared memory computer systems: SGI Origin 3800, IBM p690 (Regatta), and Compaq GS320. Considering that most of the recent shared/shared-distributed memory systems are based on the nonuniform memory access (NUMA) architecture, data *distribution* and *affinity* directives provided by each vendor are utilized to retain data locality and low cache contention (see Reference [16] for SGI Origin 3800).

LESs of turbulent flow through a straight duct are performed to evaluate the parallel performance and capability of the newly developed code (see Section 4.3 for flow-field solutions). The mesh size used for this evaluation is $257 \times 257 \times 257$ in the streamwise, vertical, and spanwise directions. Periodic boundary conditions are applied for the velocity components and pressure while, on the duct-wall, no-slip and Neumann boundary conditions are applied for the velocity and pressure, respectively. Since the present code solves Navier–Stokes equations in a generalized curvilinear coordinate system, memory and CPU time requirements are significantly higher than those of a Cartesian code usually used for a duct flow simulation.

The achieved speedups of major code parts in the computation as well as their summation on each platform are compared with a linear speedup and Amdahl's law in Figure 6. Amdahl's law is often used to estimate the parallel speedup and represents the actual limit of achievable speedup by means of the fractions of parallelized and nonparallelized parts of the code:

$$\text{Speedup}_{\text{Am}}(n) = \frac{1}{P/n + (1-P)}, \quad 0 \leqslant P < 1 \tag{38}$$

where P is obtained from $P = 2(\text{Speedup}(2) - 1)/\text{Speedup}(2)$.

The computation of R^i in Equation (21) achieves nearly linear speedups up to the maximum number of CPUs available on all three platforms. This suggests that good parallel speedup can be achieved using shared memory parallelism if memory is appropriately handled, even without significant effort put into load-balancing among the CPUs usually needed for distributed memory parallelization.

Newton iterations for the factored equation (27) and computation of the Poisson equation for pressure show deviations from linear speedup as the number of CPUs increase. The tridiagonal matrix inversions give rise to a bottleneck in the parallel performance of Newton iterations by causing a difficulty in the construction of optimal loops. On the other hand, the huge memory requirement for multigrid and resulting increased cache miss degrade the parallel speedup of the Poisson solver, which has the dominant effect on the total speedup.

Deviation from linear speedup is more significant in the Compaq GS320 which has the smallest memory capacity among the tested platforms (Figure 6(c)). The actual speedups obtained in the SGI Origin 3800 and IBM p690 (Regatta) platforms are very close to the estimations using Amdahl's law and this suggests that there is no noticeable additional overhead in the present code as the number of CPUs increases (Figures 6(a) and (b)).

4.2. LES of turbulent channel flow

Turbulent flow through a plane channel has been widely considered as a benchmark for validating numerical schemes and turbulence models. The Reynolds number is 180 based on the channel half height (δ) and friction velocity (u_τ), and the standard dynamic SGS model [20] and the

DOI: 10.1002/fld

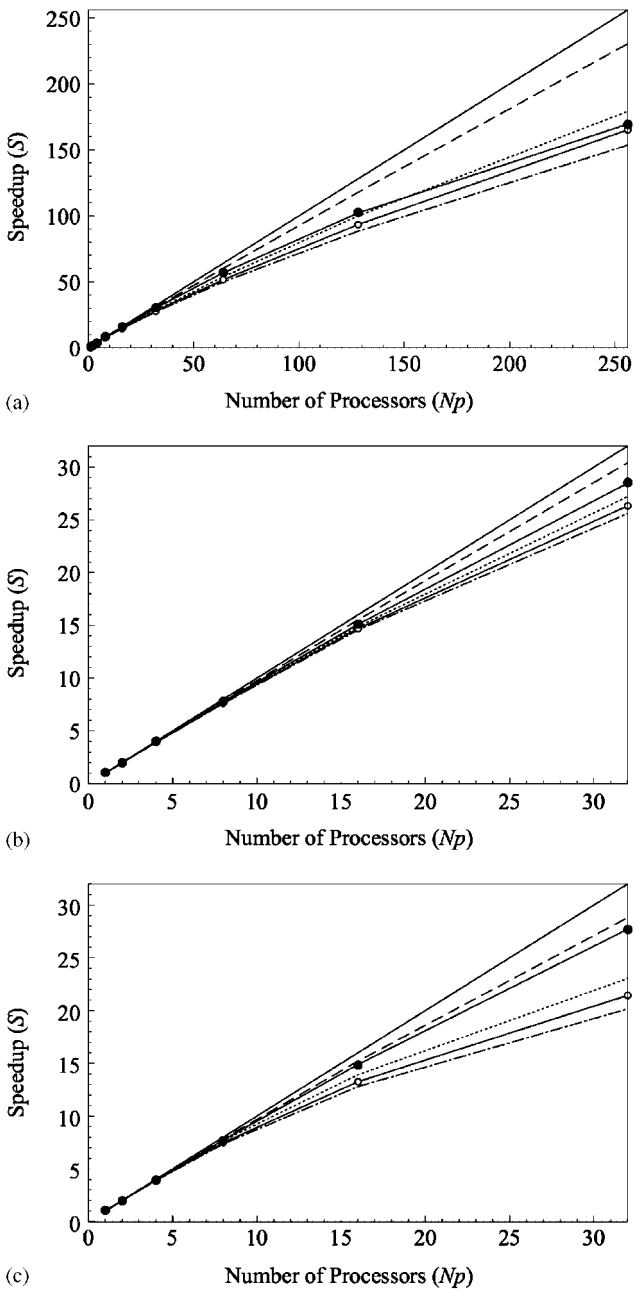

Figure 6. Parallel speedups achieved on various platforms: (a) SGI Origin 3800; (b) IBM p690 (Regatta); and (c) Compaq GS320. —, linear speedup; – – – –, RHS (Equation (21)); · · · · · ·, Newton iteration (Equation (27)); —·—, Poisson equation (Equation (28)); —●—, Amdahl's law; —○—, total speedup (summation of RHS, Newton iteration, and Poisson equation).

DOI: 10.1002/fld

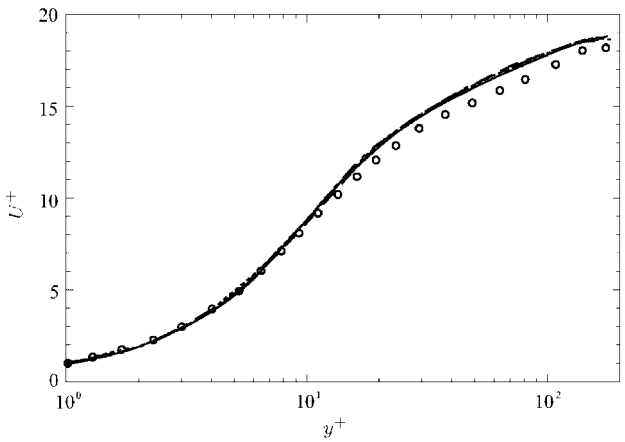

Figure 7. Mean streamwise velocity in wall units at $Re_\tau = 180$. —, LES$^{\mathrm{L}}$ with $\theta = 1.5$; · · · · · ·, LES$^{\mathrm{L}}$ with $\theta = 0.5$; —·—, LES$^{\mathrm{L}}$ with $\theta = 3.0$; – – – –, LES$^{\mathrm{D}}$; ∘, DNS [29]. Mesh size is $64 \times 96 \times 64$ ($x \times y \times z$).

Lagrangian dynamic SGS model [1] employing three different relaxation factors (θ in Equation (7)) for averaging time-scale are compared with direct numerical simulation (DNS) results [29].

A mesh of $64 \times 96 \times 64$ grid points in the streamwise, wall-normal, and spanwise directions, respectively, is employed in the computational domain size of $4\pi\delta \times 2\delta \times \frac{4}{3}\pi\delta$. Three simulations are performed with the Lagrangian dynamic model while one simulation employs the standard dynamic model.

Mean streamwise velocities and rms of velocity fluctuations from the simulations are shown in Figures 7 and 8. In those figures, LES$^{\mathrm{L}}$ and LES$^{\mathrm{D}}$ denote LESs employing the Lagrangian model and standard dynamic model, respectively. Reasonable agreements are obtained with DNS results in the mean velocities and rms of velocity fluctuations. Although all the mean streamwise velocities in these simulations predict slight overshoots in the log-layer, they compare well with each other. This indicates that the Lagrangian dynamic SGS model is reasonably robust to the choice of the relaxation factor. This result is consistent with the observation in Reference [1].

4.3. LES of square duct flow

In contrast to plane channel flows, in which there is only one inhomogeneous direction, a square duct consists of two inhomogeneous flow directions and contains interaction of turbulence structures in the vicinity of the corners. This increased flow inhomogeneity provides a good test case for developing and validating the present code.

To examine the capability and performance (see Section 4.1 for the parallel performance) of the newly developed code, LESs using a Lagrangian dynamics SGS model [1] at two Reynolds numbers of $Re_\tau = 190$ and 300, where $Re_\tau = hu_\tau/\nu$ and h and u_τ are duct half width and mid-wall friction velocity, respectively, are conducted. For the averaging time scale, the relaxation factor θ is set equal to 1.5.

Periodic boundary conditions are applied in the streamwise direction and the pressure gradient that drives the flow is adjusted dynamically to maintain a constant mass flux through the duct. The

DOI: 10.1002/fld

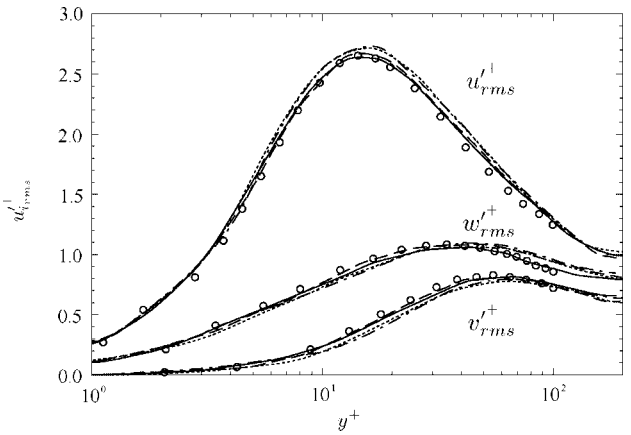

Figure 8. The rms velocity fluctuations in wall units at $Re_\tau = 180$. —, LES$^{\mathrm{L}}$ with $\theta = 1.5$; · · · · · ·, LES$^{\mathrm{L}}$ with $\theta = 0.5$; —·—, LES$^{\mathrm{L}}$ with $\theta = 3.0$; – – – –, LES$^{\mathrm{D}}$; ∘, DNS [29]. Mesh size is $64 \times 96 \times 64$ ($x \times y \times z$).

domain sizes used are $8\pi h \times 2h \times 2h$ and $4\pi h \times 2h \times 2h$ in the streamwise, vertical, and spanwise directions for $Re_\tau = 190$ and 300, respectively. The streamwise domain size is selected to be more than twice the correlation length scale in that direction.

A hyperbolic tangent function is used for distributing the grid points in cross-stream directions, and in both Reynolds number cases, 15 or more grid points are assigned below $y^+ (= yu_\tau/\nu)$, $z^+ = 10$. The resolutions in each direction are $\Delta x^+ = 29$ and 31 for $Re_\tau = 190$ and 300, respectively, and $0.15 \leqslant \Delta y^+, \Delta z^+ \leqslant 8$ for the $Re_\tau = 300$ case. A final mesh of $129 \times 129 \times 129$ used for both Reynolds numbers has been determined from a careful examination of the flow fields obtained in the simulations with $65 \times 65 \times 65$ and $97 \times 97 \times 97$ meshes. The maximum CFL number of 1, corresponding to $0.14 \leqslant \Delta t^+ = \Delta t u_\tau^2/\nu \leqslant 0.28$, is used for time integration in both Reynolds number cases.

The velocity vectors in the lower left quadrant are plotted in Figure 9. As the Reynolds number increases, the core of the secondary vortex in the lower corner-bisector tends to approach the wall and the corner. The locations of the secondary vortex cores are $({}^{y}/_{h}, {}^{z}/_{h}) = (0.20, 0.51)$ and $(0.17, 0.44)$ for $Re_\tau = 190$ and 300, respectively. They are in good agreement with the locations $(0.22, 0.52)$, $(0.24, 0.56)$, and $(0.17, 0.44)$ observed in DNS at $Re_\tau = 163$ [30], LES at $Re_\tau = 181$ [31], and DNS at $Re_\tau = 315$ [32], respectively. Note that the Reynolds numbers based on the wall-averaged friction velocities are 150, 180, and 300 in the previous duct flow studies [30–32], respectively. The mid-wall friction velocity based scaling is more appropriate for comparing the turbulence statistics along the wall-bisector [30, 32].

The mean streamwise velocities obtained from the present LES are also in reasonable agreement with those of other duct flow studies [30–32] as shown in Figure 10. The discrepancies observed in the log-law region may be caused by the different numerical schemes and resolutions used in these studies. For example, the DNS employing upwind-discretization tends to over-predict mean flow statistics in the under-resolved region [32] while the LES employing a dynamic SGS model can generate a slight overshoot in the log-layer [33].

The rms of velocity fluctuations are plotted in wall units in Figure 11 for the two Reynolds numbers in the present LES along with other duct data. Apparently, the rms velocities show

DOI: 10.1002/fld

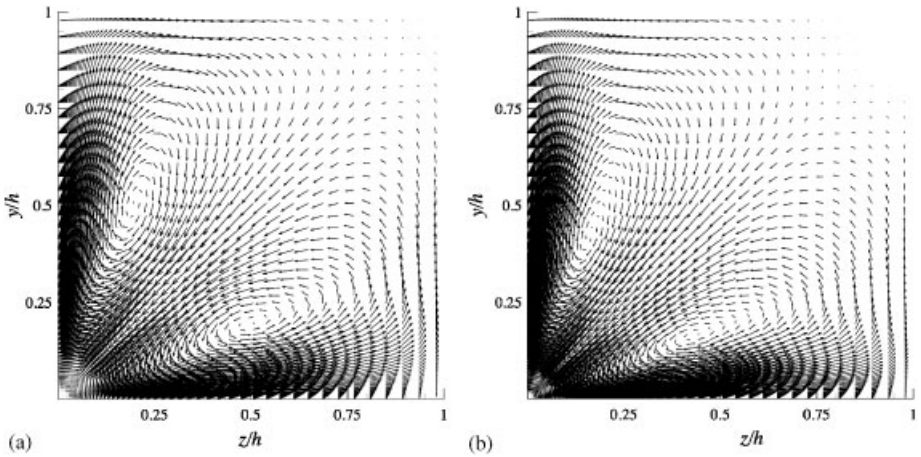

Figure 9. Mean velocity vector plots of square duct flow in a y–z plane: (a) $Re_\tau = 190$; and (b) $Re_\tau = 300$.

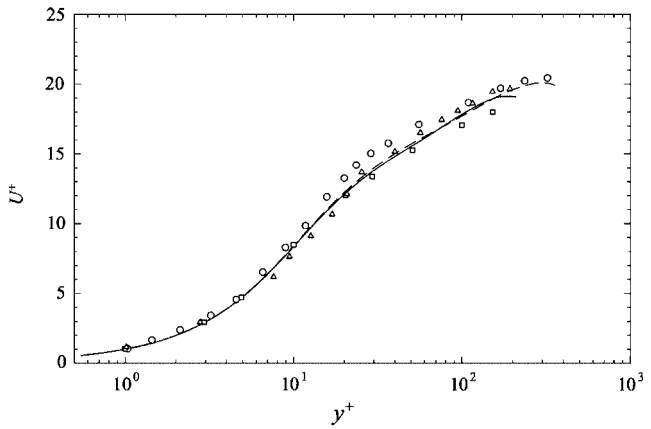

Figure 10. Mean streamwise velocity in wall units. —, present LES at $Re_\tau = 190$; – – – –, present LES at $Re_\tau = 300$; □, DNS at $Re_\tau = 163$ [30]; △, LES at $Re_\tau = 181$ [31]; ○, DNS at $Re_\tau = 315$ [32].

sensitivity to Reynolds number variations. The rms velocities obtained in the $Re_\tau = 190$ and 300 cases are in favourable agreement with other duct data for similar Reynolds numbers.

4.4. Tip-clearance flow in a turbomachinery cascade

The newly developed solver is capable of simulating the tip-clearance flow of a turbomachine, which allows an analysis of the three-dimensional flow structures as well as dynamic interactions between the tip-leakage vortex and end-wall turbulent boundary layer.

Low-Reynolds-number simulations are performed to evaluate the feasibility. The computational domain is of size $L_x \times L_y \times L_z = 1.7C \times 0.929C \times 1C$ (see Figure 12(a) for coordinate definition), where C is the chord length, and the mesh size of $321 \times 256 \times 96$ is used. To overcome the geometric difficulty in the tip gap, a novel grid topology which combines an immersed boundary

DOI: 10.1002/fld

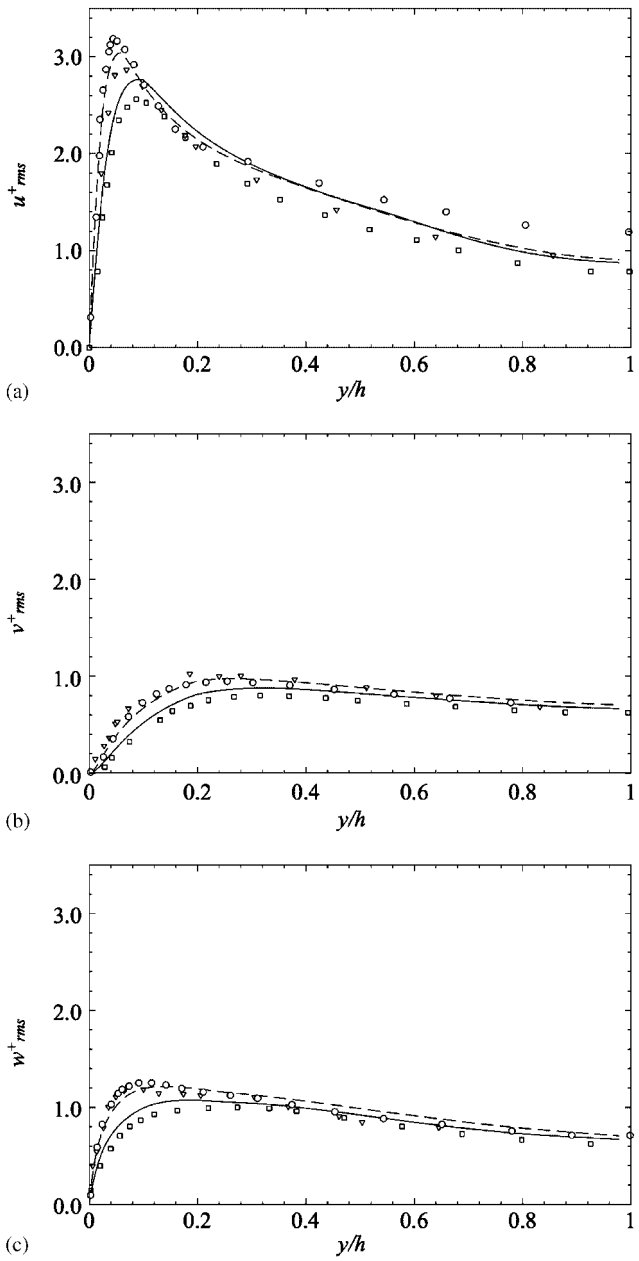

Figure 11. The rms velocity fluctuations normalized by u_τ in global coordinates: (a) u^+_{rms}, (b) v^+_{rms} and (c) w^+_{rms}. —, present LES at $Re_\tau = 190$; – – –, present LES at $Re_\tau = 300$; □, DNS at $Re_\tau = 163$ [30]; ∇, LES at $Re_\tau = 181$ [31]; ○, DNS at $Re_\tau = 315$ [32].

DOI: 10.1002/fld

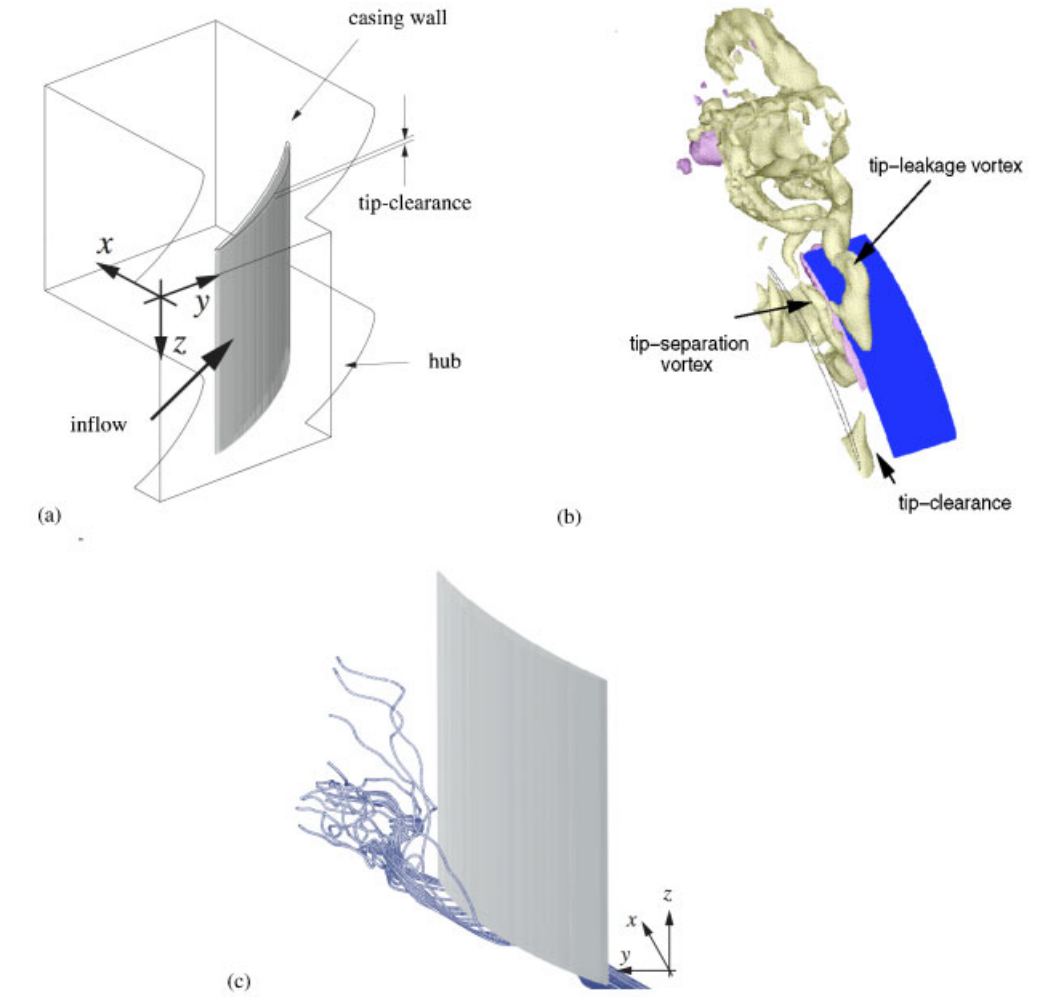

Figure 12. (a) Flow configuration and coordinate system for LES of rotor tip-leakage flow; (b) iso-pressure surfaces from a simulation at $Re_C = 3000$; and (c) streamtraces of the tip-leakage flow at $Re_C = 10\,000$.

method with a generalized curvilinear mesh has been devised (see Reference [5] for more details). Periodic boundary conditions used along the y-direction allow us to mimic the flow in the interior of a cascade. The blade has a relatively high stagger angle of about 57° and the size of the tip clearance is 1.6% of the total chord. A uniform laminar inflow and a stationary end wall are employed.

The iso-pressure surfaces of which levels are smaller than the mean pressure and the instantaneous streamtraces along the tip-leakage flow for the Reynolds numbers of 3000 and 10 000 based on the chord and inflow free-stream velocity are shown in Figures 12(b) and (c), respectively. Results from the simulations show that the method is capable of capturing the qualitative

DOI: 10.1002/fld

features of the tip-clearance flow [34]. The flow field is found to exhibit strong circular motions associated with tip-leakage and tip-separation vortices. These vortical structures are found to convect downstream, expand in size, and generate intense turbulent fluctuations in the end-wall region.

The present computational methodology is also being employed for LES of tip-clearance flow in a turbomachinery cascade with the Reynolds number of 400 000 [5, 35]. Results from the simulation show that the present LES solver is capable of capturing the complex flow features observed in the experiment [34], and allows us to probe the flow field in unprecedented detail. Qualitative and quantitative agreements with the experiment have been observed. An analysis of the velocity and pressure fields suggests a high correlation between cavitation inception and the tip-leakage vortex (see References [5, 35] for more details).

5. SUMMARY

A LES solver has been developed for high performance parallel computation of three-dimensional fully inhomogeneous flows on structured grids. Numerical algorithms and parallel strategies have been combined to obtain high computational performance and parallel scalability on the shared memory platforms.

Memory issues appear as a major bottleneck in shared-memory parallelism, especially for a large-scale computation. Strategies and algorithms to improve the memory efficiency, as well as the parallel performance of the SGS model, the factored scheme, and the Poisson solver have been devised and evaluated. The solution procedure for the Poisson equation is the most costly part in the computation of fully inhomogeneous, incompressible flow. The present combination of one-dimensionsal red–black/line Gauss–Seidel and two-dimensional red–black/tridiagonal matrix solver shows good efficiency and performance for multigrid relaxation of the Poisson equation. Also, measured global speedups are satisfactory on various shared-distributed memory systems up to the available number of CPUs on the systems.

Validations of the code have been performed in the LESs of turbulent flows through a plane channel and a square duct. Results obtained from the present solver employing a Lagrangian dynamic SGS model show good agreements with other available data. In addition, the capability of the code for fully inhomogeneous complex flows has been evaluated by performing a simulation of the tip-leakage flow in a linear cascade at moderate Reynolds numbers.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the Office of Naval Research under Grant No. N00014-99-1-0389 with Dr Ki-Han Kim as program manager. Fruitful comments of Professor Parviz Moin on a draft of this paper are greatly appreciated. Computer time was provided by a Challenge Project Grant (C82) from the DoD High Performance Computing Modernization Program (HPCMP) through Army Research Laboratory (ARL) Major Shared Resource Center.

REFERENCES

1. Meneveau C, Lund TS, Cabot WH. A Lagrangian dynamic subgrid-scale model of turbulence. *Journal of Fluid Mechanics* 1996; **319**:233–242.

DOI: 10.1002/fld

2. Ghosal S, Lund TS, Moin P, Akselvoll K. A dynamic localization model for large-eddy simulation of turbulent flows. *Journal of Fluid Mechanics* 1994; **282**:1–27.
3. Piomelli U, Liu J. Large-eddy simulation of rotating channel flows using a localized dynamic model. *Physics of Fluids* 1995; **7**(4):839–848.
4. Kim W-W, Menon S. A new incompressible solver for large-eddy simulations. *International Journal for Numerical Methods in Fluids* 1999; **31**:983–1017.
5. You D, Mittal R, Wang M, Moin P. Computational methodology for large-eddy simulation of tip-clearance flows. *AIAA Journal* 2004; **42**(2):271–279.
6. Lund TS, Wu X, Squires KD. Generation of turbulent inflow data for spatially-developing boundary layer simulations. *Journal of Computational Physics* 1998; **140**:233–258.
7. Ji S, Liu F. Flutter computation of turbomachinery cascades using a parallel unsteady Navier–Stokes code. *AIAA Journal* 1999; **37**(3):220–327.
8. Bui TT. A parallel, finite-volume algorithm for large-eddy simulation of turbulent flows. *Computers and Fluids* 2000; **29**:877–915.
9. Bartels C, Breuer M, Wechsler K, Durst F. Computational fluid dynamics applications on parallel-vector computers: computations of stirred vessel flows. *Computers and Fluids* 2002; **31**:69–97.
10. Luecke GR, Lin W-H. Scalability and performance of OpenMP and MPI on a 128-processor SGI Origin 2000. *Concurrency and Computation*: *Practice and Experience* 2001; **13**:905–928.
11. Müller M. Some simple OpenMP optimization techniques. In *WOMPAT 2001*, Eigenmann R, Voss MJ (eds). Lecture Notes in Computer Science, vol. 2104. Springer: Berlin, Heidelberg, 2001; 31–39.
12. Chapman B, Patil A, Prabhakar A. Performance oriented programming for NUMA architectures. In *WOMPAT 2001*, Eigenmann R, Voss MJ (eds). Lecture Notes in Computer Science, vol. 2104. Springer: Berlin, Heidelberg, 2001; 137–154.
13. Jia R, Sunden B. Parallelization of a multi-blocked CFD code via three strategies for fluid flow and heat transfer analysis. *Computers and Fluids* 2004; **33**:57–80.
14. Pacheco PS. *Parallel programming with MPI*. Morgan Kaufmann Publishers: San Francisco, California, 1997.
15. OpenMP Architecture Review Board, OpenMP Fortran Application Program Interface, version 2.0 Edition, http://www.openmp.org/specs, 2000.
16. Cortesi D. Origin 2000 and Onyx 2 performance tuning and optimization guide. Silicon Graphics Inc., http://techpubs.sgi.com/library/tpl/cgi-bin/init.cgi, 2001.
17. Hoeflinger J, Alavilli P, Jackson T, Kuhn B. Producing scalable performance with OpenMP: experiments with two CFD applications. *Parallel Computing* 2000; **27**:391–413.
18. Beaudan P, Moin P. Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number. *Report TF-62*, Department of Mechanical Engineering, Stanford University, Stanford, California, December 1994.
19. Mittal R, Moin P. Suitability of upwind-biased schemes for large-eddy simulation of turbulent flows. *AIAA Journal* 1997; **36**:1415–1417.
20. Germano M, Piomelli U, Moin P, Cabot WH. A dynamic subgrid-scale eddy-viscosity model. *Physics of Fluids A* 1991; **3**:1760–1765.
21. Choi H, Moin P, Kim J. Turbulent drag reduction: studies of feedback control and flow over riblets. *Report TF-55*, Department of Mechanical Engineering, Stanford University, Stanford, California, September 1992.
22. Briggs WL, Henson VE, McCormick SF. *A Multigrid Tutorial* (2nd edn). SIAM: Philadelphia, PA, 2000.
23. Zhang J. Multigrid method and fourth order compact difference scheme for 2D Poisson equation with unequal meshsize discretization. *Journal of Computational Physics* 2002; **179**:170–179.
24. Adams M, Jordan HF. Is SOR color blind? *SIAM Journal on Scientific and Statistical Computing* 1986; **7**(2): 490–506.
25. Amodio P, Mazzia F. A parallel Gauss–Seidel method for block tridiagonal linear systems. *SIAM Journal on Scientific Computing* 1995; **16**(6):1451–1461.
26. Amodio P, Mazzia F. Parallel iterative solvers for boundary value methods. *Mathematical and Computer Modelling* 1996; **23**(7):29–43.
27. Evans DJ. Parallel SOR iterative methods. *Parallel Computing* 1984; **1**(1):3–18.
28. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes*. Cambridge University Press: Cambridge, 1992.
29. Kim J, Moin P, Moser R. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics* 1987; **177**:133–166.
30. Gavrilakis S. Numerical simulation of low-Reynolds-number turbulent flow through a straight square duct. *Journal of Fluid Mechanics* 1992; **244**:101–129.

DOI: 10.1002/fld

31. Madabhushi RK, Vanka SP. Large eddy simulation of turbulence-driven secondary flow in a square duct. *Physics of Fluids A* 1991; **3**(11):2734–2744.
32. Huser A, Biringen S. Direct numerical simulation of turbulent flow in a square duct. *Journal of Fluid Mechanics* 1993; **257**:65–95.
33. Akselvoll K, Moin P. Large eddy simulation of turbulent confined coannular jets and turbulent flow over a backward facing step. *Report TF-63*, Department of Mechanical Engineering, Stanford University, Stanford, California, February 1995.
34. Wang Y, Devenport WJ. Wake of a compressor cascade with tip gap. Part 2. Effects of endwall motion. *AIAA Journal* 2004; **42**(11):2332–2340.
35. You D, Moin P, Wang M, Mittal R. Study of tip clearance flow in a turbomachinery cascade using large eddy simulation. *Report TF-86*, Department of Mechanical Engineering, Stanford University, Stanford, California, May 2004.

DOI: 10.1002/fld