CaLES

A GPU-accelerated solver for large-eddy simulation of wall-bounded flows

Xiao, Maochao; Ceci, Alessandro; Costa, Pedro; Larsson, Johan; Pirozzoli, Sergio

DOI
10.1016/j.cpc.2025.109546

Publication date
2025

Document Version
Final published version

Published in
Computer Physics Communications

Citation (APA)
Xiao, M., Ceci, A., Costa, P., Larsson, J., & Pirozzoli, S. (2025). CaLES: A GPU-accelerated solver for large-eddy simulation of wall-bounded flows. *Computer Physics Communications*, *310*, Article 109546. https://doi.org/10.1016/j.cpc.2025.109546

Computer Physics Communications 310 (2025) 109546

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computer Programs in Physics

CaLES: A GPU-accelerated solver for large-eddy simulation of wall-bounded flows

Maochao Xiao [a,*], Alessandro Ceci [a], Pedro Costa [b], Johan Larsson [c], Sergio Pirozzoli [a,*]

[a] Dipartimento di Ingegneria Meccanica e Aerospaziale, Sapienza Università di Roma, Via Eudossiana 18, 00184 Roma, Italy
[b] Process and Energy Department, TU Delft, Leeghwaterstraat 39, Delft 2628 CB, the Netherlands
[c] University of Maryland, College Park, MD 20742, USA

ARTICLE INFO

The review of this paper was arranged by Prof. Weigel Martin

Keywords:
Large-eddy simulation
GPU-accelerated solver
Fast Poisson solver
Wall-bounded turbulence
Wall model

ABSTRACT

We introduce CaLES, a GPU-accelerated finite-difference solver designed for large-eddy simulations (LES) of incompressible wall-bounded flows in massively parallel environments. Built upon the existing direct numerical simulation (DNS) solver CaNS, CaLES relies on low-storage, third-order Runge-Kutta schemes for temporal discretization, with the option to treat viscous terms via an implicit Crank-Nicolson scheme in one or three directions. A fast direct solver, based on eigenfunction expansions, is used to solve the discretized Poisson/Helmholtz equations. For turbulence modeling, the classical Smagorinsky model with van Driest near-wall damping and the dynamic Smagorinsky model are implemented, along with a logarithmic law wall model. GPU acceleration is achieved through OpenACC directives, following CaNS-2.3.0. Performance assessments were conducted on the Leonardo cluster at CINECA, Italy. Each node is equipped with one Intel Xeon Platinum 8358 CPU (2.60 GHz, 32 cores) and four NVIDIA A100 GPUs (64 GB HBM2e), interconnected via NVLink 3.0 (200 GB/s). The inter-node communication bandwidth is 25 GB/s, supported by a DragonFly+ network architecture with NVIDIA Mellanox InfiniBand HDR. Results indicate that the computational speed on a single GPU is equivalent to approximately 15 CPU nodes, depending on the treatment of viscous terms and the subgrid-scale model, and that the solver efficiently scales across multiple GPUs. The predictive capability of CaLES has been tested using multiple flow cases, including decaying isotropic turbulence, turbulent channel flow, and turbulent duct flow. The high computational efficiency of the solver enables grid convergence studies on extremely fine grids, pinpointing non-monotonic grid convergence for wall-modeled LES.

Program summary

Program title: CaLES
CPC Library link to program files: https://doi.org/10.17632/6chjn6zdmz.1
Developer's repository link: https://github.com/soaringxmc/CaLES
Licensing provisions: MIT License
Programming language: Fortran 90, OpenACC, CUDA Fortran, MPI
Nature of problem: Direct numerical simulation (DNS) and large-eddy simulation (LES) of incompressible wall-bounded turbulent flows. The program is designed for a variety of flow configurations, including channel flow, duct flow, cavity flow, etc.
Solution method: The filtered Navier-Stokes equations are solved using a fractional-step method. Time integration is performed with a low-storage, three-step Runge-Kutta scheme, while spatial discretization employs a second-order finite-difference method on staggered grids. The coupling between the pressure and velocity fields is achieved through a pressure-correction method. The program incorporates subgrid-scale (SGS) modeling, featuring the classical Smagorinsky model and its dynamic variant, along with wall modeling based on the classical logarithmic law.

1. Introduction

Large-eddy simulation (LES) has become an important tool in the design processes of spacecraft, aircraft, automobiles, ships, and other engineering systems, for cases where traditional Reynolds-averaged Navier-Stokes (RANS) models fall short. RANS models often struggle to accurately predict complex flow phenomena, such as flow separation, rotational turbulence, and three-dimensional turbulent boundary lay-

* Corresponding authors.
E-mail addresses: maochao.xiao@uniroma1.it (M. Xiao), sergio.pirozzoli@uniroma1.it (S. Pirozzoli).

https://doi.org/10.1016/j.cpc.2025.109546
Received 20 November 2024; Received in revised form 3 February 2025; Accepted 11 February 2025

Available online 14 February 2025

ers [1]. Direct numerical simulation (DNS) can provide accurate results but become computationally prohibitive at high Reynolds numbers due to the high mesh resolutions required. As a result, LES has gained popularity for offering a reasonable balance between computational cost and predictive accuracy, particularly for complex aerodynamic and hydrodynamic applications. In recent years, LES has been increasingly applied to external aerodynamic flows, especially at the edges of flight envelopes, such as high-lift aircraft aerodynamics [2] and iced-wing separated flows [3]. It has also been used in internal flows, including those within aircraft engines [4]. These applications establish LES as a critical component in the industry push toward Certification by Analysis [5]. Despite significant advancements, both wall models and subgrid-scale (SGS) closures for LES remain areas of active development. For wall modeling, researchers have been exploring robust models that account for more complex flows, such as laminar-turbulent transitions [6] and flow separation [7] among others. Concurrently, advancements in SGS models have been made, with some of the recent studies emphasizing robust SGS models suited to anisotropic grids [8].

It is well established that LES is significantly more computationally demanding than RANS, making the development of faster LES solvers a critical need. An efficient LES solver is not only essential for large-scale production simulations, but also invaluable for the development and testing of subgrid-scale and wall models. Indeed, as data-driven approaches and machine learning techniques are increasingly applied to turbulence modeling, the demand for efficient LES solvers becomes even more pressing. Previous studies have trained SGS models using filtered DNS data [9,10]. However, relying on DNS data can be problematic, as the SGS tensor in implicitly filtered LES does not perfectly align with the SGS stress terms derived from the filtered Navier-Stokes equations [11]. This inconsistency is especially significant when grid sizes are significantly larger than the Kolmogorov scale, as the numerical and modeling errors can then be comparable. As a result, SGS models trained on filtered DNS data may perform well in *a priori* tests but fail in *a posteriori* assessments, whereby performance is evaluated in actual LES simulations. Consequently, research has increasingly shifted toward generating training data directly from LES, and turbulence models are optimized for accurate statistical metrics such as mean velocity and wall shear stress [12]. In the same vein, reinforcement learning has been explored for both SGS and wall modeling [13,14]. Such strategies have been referred to as "model-consistent" approaches [15]. While these strategies show promise, they often require hundreds or even thousands of LES runs to generate training data or complete one-time training, underscoring the need for fast and efficient solvers. Moreover, the grid convergence properties of wall-modeled LES (WMLES) have become increasingly studied [16,17], which also requires many LES runs on a series of refined meshes for thorough analysis. Fast LES solvers are therefore desirable to enable grid convergence investigations at high Reynolds numbers.

Table 1 gives some popular open-source solvers used in academic research. Whereas all those solvers are capable of performing DNS for canonical flow cases, only URANOS and LESGO currently support LES capabilities. It is well-known that incompressible solvers, such as LESGO, are typically more efficient than compressible solvers like URANOS for simulations of low-Mach-number flows. This efficiency mainly derives from allowing much larger time steps when explicit time-integration schemes are used. However, a key limitation of LESGO is the absence of GPU acceleration, which restricts its scalability and efficiency on modern high-performance computing platforms. Given the high demand for LES in simulating incompressible flows, particularly for machine-learning-based turbulence modeling and grid convergence studies, the development of a GPU-accelerated incompressible LES solver is highly desirable in academia.

The present work introduces CaLES, a GPU-accelerated incompressible LES solver specifically designed for wall-bounded flows. CaLES builds on the capabilities of CaNS [22,23], an open-source DNS solver known for its efficiency in solving the incompressible Navier-Stokes equations. The solver uses a fractional-step method [24]. Temporal discretization is carried out using a low-storage, third-order Runge-Kutta scheme [25]. Spatial discretization is performed using a second-order finite-difference method on staggered grids [26], which avoids odd-even decoupling phenomena and preserves energy at the discrete level in the inviscid limit [27]. The solver employs eigenfunction expansions [28] to efficiently solve the Poisson equation. GPU acceleration is achieved using a combination of CUDA Fortran and OpenACC directives, and performance benchmarks demonstrate that the code performance on 4 NVIDIA Tesla V100 GPUs in a DGX-2 system is roughly equivalent (0.9 times slower to 1.6 times faster) to 2048 cores on state-of-the-art CPU-based supercomputers, and 3.1 to 5.6 times faster when all 16 GPUs in the DGX-2 cluster are used [22]. CaLES extends CaNS by supporting LES through the inclusion of the classical Smagorinsky model [29] with the van Driest damping function [30], and the dynamic Smagorinsky model [31,32], along with a logarithmic-law wall model. The solver can simulate various canonical flows in Cartesian single-block domains, including isotropic turbulence, temporally-evolving turbulent boundary layers, channel flows, duct flows, cavity flows, etc. Flexibility and efficiency make it an ideal platform to develop subgrid-scale and wall models, particularly those based on machine learning techniques, and to perform grid convergence studies at high Reynolds numbers. In this work, we present CaLES as an extension of CaNS, which was designed to maintain simplicity and adaptability, facilitating its use across diverse applications.

Table 1
Open-source DNS/LES solvers for academic research.

Solver	Governing equations	GPU-supported	Purposes
STREAmS-2 [18]	Compressible NS	Yes	DNS
URANOS [19]	Compressible NS	Yes	DNS/LES
AFiD [20]	Incompressible NS	Yes	DNS
LESGO [21]	Incompressible NS	No	DNS/LES
CaNS [22]	Incompressible NS	Yes	DNS
CaLES	Incompressible NS	Yes	DNS/LES

The remainder of this paper is organized as follows. Section 2 introduces the governing equations, subgrid-scale models, and numerical methods. Section 3 discusses the implementation and performance of the solver. Section 4 validates the LES capabilities using decaying isotropic turbulence, turbulent channel flow, and turbulent duct flow. Finally, Section 5 provides the conclusions of this study.

2. Methodology

2.1. Governing equations and subgrid-scale models

The filtered incompressible Navier-Stokes (NS) equations read as

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0, \tag{1}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j}, \tag{2}$$

where $\overline{u}_i$ represents the filtered velocity, ρ is the fluid density, $\overline{p}$ is the filtered pressure, and ν denotes the kinematic viscosity. The term $\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j$ is the subgrid-scale stress tensor, which encapsulates the effects of unresolved scales and requires a suitable closure model. The isotropic component of the SGS stress is typically absorbed into the pressure, resulting in a modified pressure field, $\overline{p} \leftarrow \overline{p} + \frac{1}{3}\rho\tau_{kk}$. The remaining deviatoric part is modeled, according to the Boussinesq assumption, as

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\nu_t \overline{S}_{ij}, \tag{3}$$

where δ_{ij} denotes the Kronecker delta, and $\overline{S}_{ij}$ is the filtered strain-rate tensor. We implemented two representative closure models: the classical

Smagorinsky model [29] and its dynamic version [31,32]. The baseline Smagorinsky model reads as [29]

$$\nu_t = \left(C_s \Delta D(y)\right)^2 \overline{S}, \tag{4}$$

where C_s is a model constant, Δ is the filter width, $D(y)$ is the near-wall damping function, and $\overline{S}$ is the rate-of-strain, i.e., $\overline{S} = \sqrt{2\overline{S}_{ij}\overline{S}_{ij}}$. The van Driest damping function [30] is commonly used in the presence of no-slip walls, i.e., $D(y) = 1 - \exp(-y^*/25)$, where "*" denotes the wall distance non-dimensionalized by the wall units. A drawback of the standard Smagorinsky model is its inaccurate prediction of the eddy dissipation in laminar and transitional flows, leading to erroneous wall shear stresses and delayed transition to turbulence. Moreover, the optimal value of the model constant C_s depends significantly on the flow features. Lilly [33] showed that for isotropic turbulence, with spatial resolution within the inertial subrange, $C_s \approx 0.17$, whereas Deardorff [34] suggested $C_s \approx 0.1$ for wall-bounded turbulent shear flows. The Smagorinsky model can be improved using a dynamic procedure, whereby the model coefficient is evaluated dynamically by comparing the eddy dissipation at two filter levels. The dynamic Smagorinsky model with Lilly's modification [31,32] is expressed as:

$$\nu_t = c_s \Delta^2 \overline{S}, \tag{5}$$

where

$$c_s = \frac{\langle M_{ij} L_{ij} \rangle}{\langle M_{ij} M_{ij} \rangle}, \tag{6}$$

$$L_{ij} = \widetilde{\overline{u}_i \overline{u}_j} - \widetilde{\overline{u}}_i \widetilde{\overline{u}}_j, \tag{7}$$

$$M_{ij} = 2\Delta^2 \widetilde{\overline{S}\,\overline{S}_{ij}} - 2(\alpha\Delta)^2 \widetilde{\overline{S}}\,\widetilde{\overline{S}}_{ij}. \tag{8}$$

Here, the bar denotes filtering with filter width Δ, the tilde indicates test filtering with filter width $\alpha\Delta$, and the brackets $\langle\rangle$ represent an averaging operation. The ratio of the two filter widths is commonly set to $\alpha = 2.0$. The test-filtered strain rate is computed as

$$\widetilde{\overline{S}}_{ij} = \frac{1}{2}\left(\frac{\partial \widetilde{\overline{u}}_i}{\partial x_j} + \frac{\partial \widetilde{\overline{u}}_j}{\partial x_i}\right). \tag{9}$$

The dynamic Smagorinsky model provides reasonable subgrid-scale dissipation and automatically switches off in laminar flows. However, it demands more memory and computational resources than the static version and often requires averaging or clipping to ensure numerical stability [32].

Wall models are required when the near-wall mesh resolution is too coarse to resolve the inner-layer turbulent scales up to the inertial subrange. These models are typically classified into near-wall RANS models [35] and wall-stress models [36]. In classical wall-stress models, the wall-parallel velocity at a specific wall distance is used as input and the wall shear stress is estimated as output, which replaces the no-slip boundary condition. The simplest wall-stress model is the logarithmic wall law:

$$\frac{U_{wm}}{u_\tau} = \frac{1}{\kappa}\ln\left(\frac{u_\tau h_{wm}}{\nu}\right) + B, \tag{10}$$

where $\kappa = 0.41$, $B = 5.2$, u_τ is the friction velocity, h_{wm} is the wall-modeled layer thickness, and U_{wm} is the wall-parallel velocity magnitude at the top of the wall-modeled layer. At first sight, the logarithmic wall model may appear to be accurate only for equilibrium flows, but both experience and basic near-wall turbulence physics suggest that this is not generally the case [6]. A high-quality WMLES will resolve the dynamically important eddies in the outer layer, and should therefore accurately capture non-equilibrium effects there. The turbulence time scale decreases towards the wall in the inner layer, and thus the turbulence there may well exist in some type of quasi-equilibrium even in a non-equilibrium outer flow. The literature contains some evidence supporting this, especially when focusing on studies in which the WMLES grid and wall-model exchange location were carefully designed to limit those errors. For example, Bermejo-Moreno et al. [37] conducted WMLES using an equilibrium wall model to study interactions between oblique shock waves and turbulent boundary layers in a nearly square duct. Despite having secondary corner flows and shock-induced three-dimensional separation bubbles, their results showed favorable agreement with experimental data.

2.2. Numerical methods

The filtered incompressible Navier-Stokes equations are solved using a fractional-step method [24]. Time integration is performed with a low-storage, three-step Runge-Kutta scheme [25], while spatial discretization is handled using a second-order finite-difference method on staggered grids [26]. The continuity equation (1) and the momentum equation (2) are coupled through a pressure-correction method [38]. This section provides an overview of the numerical schemes, with a focus on key aspects of the implementation of SGS and wall models. For additional details, refer to the descriptions of CaNS [23].

At each sub-step, the flow field is updated as

$$u_i^* = u_i^k + \Delta t\left[\alpha_{\kappa+1}\left(H_i^k + \nu L_{jj} u_i^k\right) + \beta_{\kappa+1}\left(H_i^{k-1} + \nu L_{jj} u_i^{k-1}\right) - \gamma_{\kappa+1}\partial_i p^{k-1/2}\right], \tag{11}$$

$$L_{jj}\phi = \frac{\partial_i u_i^*}{\gamma_{k+1}\Delta t}, \tag{12}$$

$$u_i^{k+1} = u_i^* - \gamma_{k+1}\Delta t \partial_i \phi, \tag{13}$$

$$p^{k+1/2} = p^{k-1/2} + \phi. \tag{14}$$

Here, $k = 0$ corresponds to physical time level n, and $k + 1 = 3$ corresponds to time level $n + 1$. The symbol u_i^* denotes the predicted velocity, and ϕ represents the pressure correction. The Runge-Kutta coefficients are $\alpha_{k+1} = (8/15, 5/12, 3/4)$, $\beta_{k+1} = (0, -17/60, -5/12)$, and $\gamma_{k+1} = \alpha_{k+1} + \beta_{k+1}$. In equation (11), H_i includes convective and SGS stress terms,

$$H_i = \frac{\partial u_i u_j}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_j}, \tag{15}$$

and the Laplace operator in the diffusive term is defined as

$$L_{jj} = \frac{\partial^2}{\partial x_j \partial x_j}. \tag{16}$$

For low-Reynolds-number flows or very fine grids, it may be desirable to use implicit temporal discretization for the diffusion terms. With Crank-Nicolson time integration, this results in

$$u_i^{**} = u_i^k + \Delta t\left[\alpha_{\kappa+1} H_i^k + \beta_{\kappa+1} H_i^{k-1} + \gamma_{\kappa+1}\left(\nu L_{jj} u_i^k - \partial_i p^{k-1/2}\right)\right], \tag{17}$$

$$u_i^* - \gamma_{\kappa+1}\frac{\nu\Delta t}{2} L_{jj}\left(u_i^*\right) = u_i^{**} - \gamma_{\kappa+1}\frac{\nu\Delta t}{2} L_{jj} u_i^k, \tag{18}$$

$$p^{k+1/2} = p^{k-1/2} + \phi - \gamma_{k+1}\frac{\nu\Delta t}{2} L_{jj}\phi. \tag{19}$$

Equations (17) and (18) are intentionally not combined to illustrate that u_i^{**} provides a better approximation of u_i^{k+1} than the sum of the terms on the right-hand side of equation (18) [22]. Alternatively, implicit treatment of the viscous terms can be performed only in the y direction, resulting in

$$u_i^{**} = u_i^k + \Delta t\left\{\alpha_{k+1}\left[H_i^k + \nu\left(L_{11} + L_{33}\right)u_i^k\right] + \beta_{k+1}\left[H_i^{k-1} + \nu\left(L_{11} + L_{33}\right)u_i^{k-1}\right] + \gamma_{k+1}\left(\nu L_{22} u_i^k - \partial_i p^{k-1/2}\right)\right\}, \tag{20}$$

$$\left(1 - \gamma_{\kappa+1}\frac{\nu\Delta t}{2} L_{22}\right) u_i^* = u_i^{**} - \gamma_{\kappa+1}\frac{\nu\Delta t}{2} L_{22} u_i^k, \tag{21}$$

$$p^{k+1/2} = p^{k-1/2} + \phi - \gamma_{k+1}\frac{\nu\Delta t}{2} L_{22}\phi. \tag{22}$$

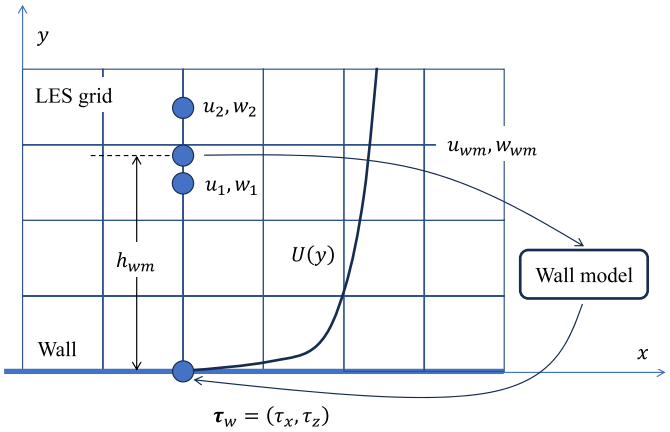


Fig. 1. Schematic of the wall-model implementation.

In the solver, z is designated as the coordinate direction along which non-uniform grid spacing can be applied, typically corresponding to the wall-normal direction in a plane channel. However, to maintain consistency with the conventions in the turbulence community, we refer to the wall-normal direction as y throughout this paper. The Poisson equation (12), once discretized, is solved using an eigenfunction expansion method [28], which allows for an efficient solution using the Thomas algorithm along the y direction. When implicit treatment of the viscous diffusive terms is applied, the resulting three modified Helmholtz equations in equation (18) are solved with the same direct solver used for the Poisson equation.

The SGS stress terms are always handled explicitly, regardless of the time integration scheme used for the viscous terms. This allows the SGS stress terms to be grouped with the convective terms, as shown in equation (15). The SGS model is evaluated at cell centers, where both the SGS viscosity and the strain-rate tensor are stored. The diagonal components of the strain-rate tensor are directly computed at the cell centers, while the non-diagonal components are first calculated at the cell edge midpoints and then averaged to the cell centers. In the dynamic procedure, two-dimensional (2D) or three-dimensional (3D) box filters can be applied. The implementation of the filters assumes uniform grid spacing for simplicity. The 3D box filter cannot be used in the first off-wall layer, hence a 2D box filter is applied through linear extrapolation of the wall-parallel velocity to the ghost points, followed by the application of a 3D filter. In this layer, we set $\alpha = 4^{1/3}$ in equation (8), which mathematically corresponds to a 2D box filter. Our WMLES tests on channel flows indicate that using this value in this first layer yields more accurate results than $\alpha = 8^{1/3}$. The averaging operation in equation (6) is performed in the homogeneous directions, and the averaged coefficient is clipped to zero when it is negative. The velocity components are averaged to the cell centers before applying the test filter to evaluate L_{ij} in equation (7).

For the wall model, the Newton–Raphson iterative method is used to determine the wall shear stress from equation (10), typically requiring 3 to 7 iterations to achieve convergence in wall shear stress within a relative tolerance of 0.01%. Fig. 1 illustrates how the wall model is coupled to the LES solution. The input velocity at the top of the wall-modeled layer is the magnitude of the instantaneous wall-parallel velocity, which is evaluated as $U_{wm} = (u_{wm}^2 + w_{wm}^2)^{1/2}$, where u_{wm} and w_{wm} are the x- and z-direction velocities, respectively. The two components are obtained via linear interpolation in the wall-normal direction between locations 1 and 2. The resulting wall shear stress is then used as the boundary condition for the wall-parallel velocity vector. The impermeability condition is enforced for the wall-normal velocity component. When a staggered grid is used, the two wall-parallel velocity components (e.g., u and w) are stored at different cell face centers. Consequently, when the wall model is used to enforce the boundary condition for u, as illustrated in Fig. 1, interpolation of w to the location of u is required to evaluate the wall-parallel velocity vector. Similarly, for w, interpolation of u to the location of w is performed. This approach minimizes the number of interpolations, which is desirable when computing either SGS viscosity or wall models on staggered grids. When utilizing a wall model, the wall-normal derivatives of the wall-parallel velocity components are determined from first-order one-sided finite-difference scheme in the first off-wall layer, thus avoiding crossing the under-resolved layer between the wall and the first off-wall wall-parallel velocity location [36]. In CaLES, one-sided finite differencing is achieved through linear extrapolation of the wall-parallel velocity to the ghost points, followed by second-order central differencing. This procedure is crucial for accurate evaluation of the strain-rate tensor required to evaluate the SGS viscosity. The viscous terms in the filtered Navier-Stokes equations are evaluated as usual in the first off-wall layer, as the wall shear stress is directly provided by the wall model, and the layer between the first and second off-wall locations of the wall-parallel velocity can be regarded as properly resolved. When the viscous terms are handled implicitly in all three directions, the wall model can only be applied in the y-direction, as homogeneous boundary conditions are required in the other two directions where Fourier transforms are applied. However, this limitation may be irrelevant, as explicit time integration is commonly used for wall-modeled LES due to the large thickness of the first off-wall layer of cells. This restriction does not apply when the viscous terms are handled implicitly only in the y-direction.

3. Implementation and performance

3.1. Overall implementation strategy

CaLES is developed using CaNS-2.3.0 as its baseline solver. Algorithm 1 outlines the overall solution procedure for explicit time integration. When the viscous terms are handled implicitly, u_i^* is computed using equations (17) and (18), and $p^{k+1/2}$ is updated from equation (19). When the viscous terms are handled implicitly only in the y-direction, u_i^* is determined by equations (20) and (21), while $p^{k+1/2}$ is updated from equation (22). To ensure consistency with boundary conditions, ghost cells are updated immediately after any variable is updated. Algorithm 2 details the procedure for applying boundary conditions. The calculation of wall shear stress using the wall model must be performed after all other boundary conditions have been applied, as the wall model computation relies on the updated ghost-point values at boundary points.

Algorithm 1 Overall solution procedure.

1: Initialize velocity u_i and pressure p.
2: Compute eddy viscosity ν_t.
3: Set iteration counter $n = 0$.
4: **while** $n \leq n_{\max}$ **do**
5: Increment iteration counter: $n \leftarrow n + 1$.
6: Determine time step Δt.
7: **for** $k = 0$ to 2 **do**
8: Compute intermediate velocity u_i^* using equation (11);
9: Compute the right-hand side of the Poisson equation and solve for ϕ in equation (12);
10: Update u_i^{k+1} using the correction procedure of equation (13);
11: Update $p^{k+1/2}$ using equation (14);
12: Compute ν_t^{k+1}.
13: **end for**
14: **end while**
15: Terminate simulation.

Algorithm 2 Boundary condition treatment.

1: Perform ghost-cell exchange between blocks.
2: Update all boundary conditions except wall-model boundary conditions.
3: Compute wall shear stress using equation (10).
4: Update wall-model boundary conditions.

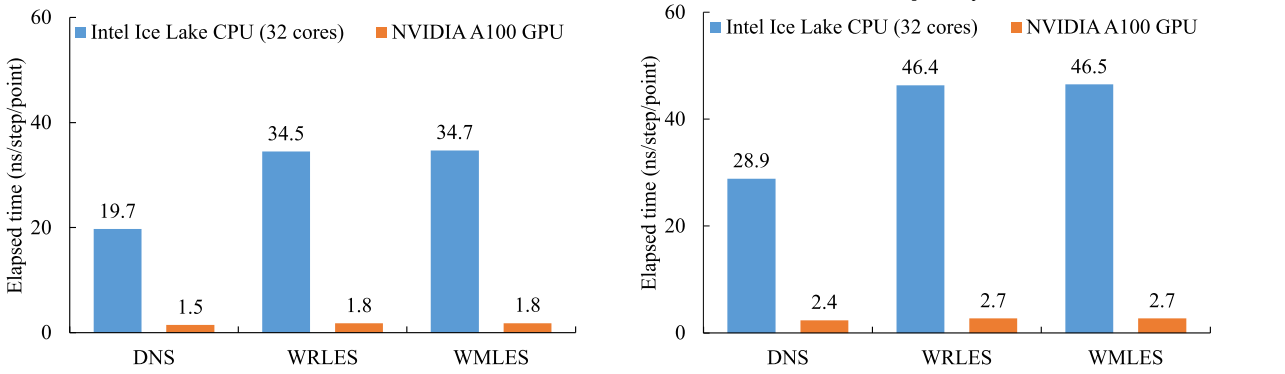


Fig. 2. Elapsed wall-clock time per time step per grid point for different methods when the viscous terms are handled explicitly (a), or implicitly in the y direction (b).

The GPU porting was implemented using OpenACC directives. A key factor to achieving high acceleration is to minimize data transfer between GPU and CPU memory. In the solver, large arrays are transferred only during the initialization or the first time step using unstructured data lifetimes. From the second iteration onward, only scalars or small arrays are transferred between CPU and GPU memory, except when flow-field output is required. According to the OpenACC Programming and Best Practices Guide [39], parallel regions are defined using either the "kernels" construct or the"parallel" construct. Specifically, the "kernels" construct allows the compiler to automatically exploit parallelism in a region of code, while the "parallel" construct, often used in conjunction with the "loop" construct and the "collapse" clause, is applied to accelerate key loops for optimal performance. We use the "kernels" construct for simple array assignments, and the "parallel" construct to speed up loops for the computation of advective, viscous and SGS stress terms, as well as for time advancement. Additionally, the "async" clause is applied to the kernels, parallel, update, and data directives (both structured and unstructured), enabling the CPU to continue with other tasks while the accelerator performs operations, without waiting for their completion. The FFTs required by the Poisson/Helmholtz solver are carried out using the cuFFT library from the CUDA Toolkit. For additional details, refer to [22].

In line with CaNS-2.3.0, parallelization of the code is achieved through MPI, with each rank allocated to one GPU. The structured grid block is partitioned into subdomains using a 2D pencil-like decomposition. The pencil axis is recommended to be aligned with the x-direction for optimal efficiency, except when viscous terms are handled implicitly in the y-direction, in which case alignment with the y-axis is preferable. The cuDecomp library [40] manages the transpose operations required for FFT-based transforms and ghost-cell exchanges. The library not only optimizes the pencil domain decomposition layout, but also finds the most efficient communication backends for transposes and ghost-cell exchanges. This process involves runtime testing of different grid decomposition layouts and communication backends to identify the best-performing combination. Notably, transpose operations and ghost-cell exchanges can utilize different communication backends. Supported communication methods include CUDA-aware MPI point-to-point, MPI all-to-all, NVIDIA Collective Communication Library (NCCL), and NVIDIA Shared Memory (NVSHMEM), with different staging strategies. Given that the optimal setup is highly system-dependent, the hardware-adaptive decomposition provided by cuDecomp is crucial for efficient resource utilization [40].

3.2. Performance analysis

Assessment of the code performance was conducted on the Booster partition of the Leonardo cluster at CINECA, Italy. Each node of the cluster is equipped with an Intel Xeon Platinum 8358 CPU (2.60 GHz, 32 cores) and four NVIDIA A100 GPUs (64 GB HBM2e). The intra-node communication bandwidth is 200 GB/s, supported by NVLink 3.0. The inter-node bandwidth is 25 GB/s, facilitated by the DragonFly+ network architecture using NVIDIA Mellanox InfiniBand HDR, giving each GPU an effective communication rate of approximately 6.25 GB/s. The performance comparison is made between a single GPU card and a CPU node. The test case under consideration is flow in a plane channel with two walls in the y direction, and periodic boundary conditions in the other two directions. The grid size is $(N_x, N_y, N_z) = (512, 384, 1440)$, which is approximately the maximum grid size that can fit into GPU memory when the Smagorinsky model with the van Driest damping function is activated and the viscous terms are handled implicitly along the y direction. It is noteworthy that, during runs on a single GPU, data sharing is also performed when handling periodic boundary conditions in the two decomposed directions, with the pencil-axis as the non-decomposed direction. Fig. 2 compares the wall-clock time for different methods when the viscous terms are handled explicitly or implicitly in the y direction. The inclusion of the wall model results in negligible computational overhead. However, applying the static Smagorinsky model with the van Driest damping function increases the computational cost by approximately 0.3 ns per step per grid point. When the viscous terms are handled explicitly, the speed-up factors relative to the CPU node are 13× for DNS and 19× for wall-resolved LES (WRLES). If the viscous terms are handled implicitly in the y direction, the speed-up factors are 12× for DNS and 17× for WRLES. The greater speed-up of WRLES is due to the increased computational intensity introduced by the subgrid-scale model.

Table 2
Estimated memory footprint per grid point for different methods. "SM" denotes the Smagorinsky model with the van Driest damping function, and "DSM" is the dynamic Smagorinsky model.

Method	DNS	WRLES (SM)	WRLES (DSM)
Explicit	136 bytes	168 bytes	352 bytes
Implicit-y	160 bytes	192 bytes	376 bytes

Fig. 3 reports the wall-clock time breakdown for different computational components. The most time is consumed by the Poisson solver and the computation of the right-hand side of the momentum equation. The calculation of eddy viscosity using the Smagorinsky model with the van Driest damping function is the third most time-consuming part. When the viscous terms are handled implicitly in the y direction, the Helmholtz solver is the third most time-consuming component, followed by the eddy viscosity calculation. Table 2 presents the estimated memory footprint per grid point for different methods. The incorporation of the Smagorinsky model with the van Driest damping function increases memory usage per grid point by 32 bytes, or approximately 20% of the DNS memory footprint. In contrast, the dynamic Smagorinsky model requires an additional 216 bytes, or about 150% of the DNS

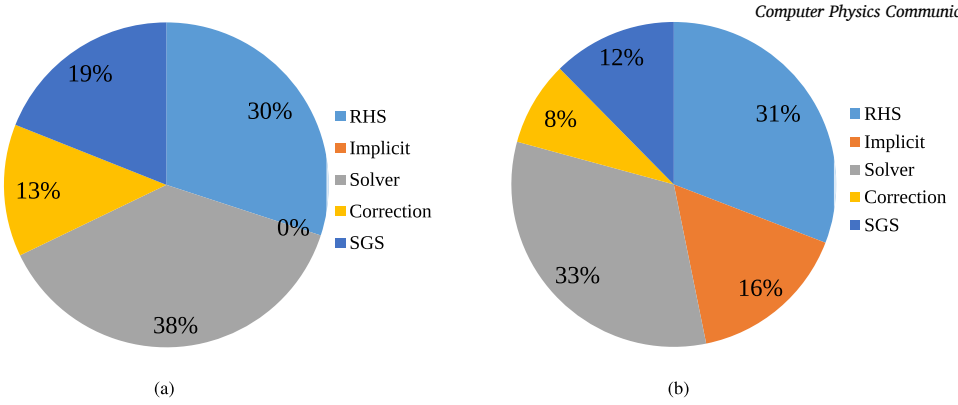


Fig. 3. Comparison of wall-clock time breakdown for different computational components, when the viscous terms are handled explicitly (a), or implicitly in the y direction (b). "RHS" denotes the time spent calculating the right-hand side of the momentum equation, given the eddy viscosity; "Implicit" indicates the time required to solve the Helmholtz equations when the viscous terms are handled implicitly in the y direction; "Solver" corresponds to the time spent solving the Poisson equation; "Correction" denotes the time allocated for the correction procedure; and "SGS" represents the time required to evaluate the eddy viscosity using the classical Smagorinsky model with the van Driest damping function.

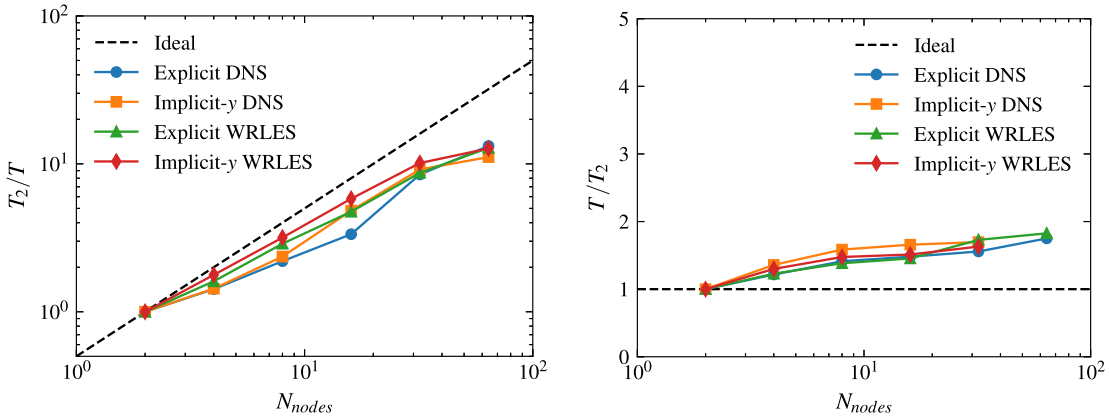


Fig. 4. Strong (a) and weak (b) code scaling performance. Here, N_{nodes} denotes the number of nodes, T_2 represents the wall-clock time for $N_{nodes} = 2$, and T is the wall-clock time for a given N_{nodes}.

memory footprint. Note that memory requirements for WMLES simulations exactly match those of the corresponding WRLES cases and are therefore omitted from the table.

We now proceed to analyze the strong and weak scalability performance of the solver. For the strong scaling experiments, a fixed grid size of $(N_x, N_y, N_z) = (512, 384, 1440 \times 4 \times 2)$ is used. This grid is generated by refining the previous single GPU grid by a factor of 8 along the spanwise z direction, resulting in a size close to the maximum allowable grid on two nodes when the viscous terms are handled implicitly in the y direction with the Smagorinsky model and the van Driest damping function activated. For the weak scaling tests, the grid size was set to $(512, 384, 1440 \times 4 \times N_{nodes})$. We begin the scaling tests from two nodes instead of one, to avoid performance degradation caused by transitioning from fast intra-node communication to slower inter-node communication. The wall-clock time for two nodes, T_2, is used as the reference for reporting the code scaling performance. The domain decomposition is aligned with the x-axis when the viscous terms are handled explicitly, and with the y-axis when the viscous terms are handled implicitly in the y direction. Fig. 4 presents the results for both strong and weak scaling. In GPU-resident, distributed-memory simulations of turbulent flows, weak scaling is the most critical performance metric. Maximizing GPU occupancy is always desirable, making it essential that the code maintains high performance for a fixed problem size per computational subdomain or MPI task, and consequently per GPU. Fig. 4(b) demonstrates that the weak scaling performance across the different computational configurations has small variations. The computational time increases by approximately 80% as the number of nodes increases from 2 to 64. When the viscous terms are handled implicitly in the y direction, there are no data points for $N_{nodes} = 64$ due to GPU memory limitations. The fully implicit scheme is not analyzed here due to its low computational efficiency when Fourier transforms are applied to solve the momentum equations. In such cases, more efficient techniques, such as the alternating-direction-implicit (ADI) scheme [24], may be employed due to its $O(N)$ computational complexity for a single dimension and its additional options for parallel implementation. By contrast, FFT methods have $O(N \log N)$ complexity and typically require extensive data transpositions. However, it should be noted that ADI can still be computationally expensive in a distributed-memory setting, as it requires either transposing the domain or solving a sequence of three tri-diagonal systems in all three directions.

4. Validation

The solver is validated using three representative cases: homogeneous decaying isotropic turbulence (DIT), turbulent channel, and turbulent duct flow. The DIT case is employed to validate the SGS models, while the channel case at $Re_b = 20,000$ is to validate the wall-resolved LES capability. The channel flow at $Re_b = 250,000$ and the duct flow are instead employed to validate the wall-modeled LES capability. In these simulations, the viscous terms are handled explicitly, except for the wall-resolved LES of the channel flow, where they are handled implicitly in the y direction. For all simulations, the time step is dynamically adjusted

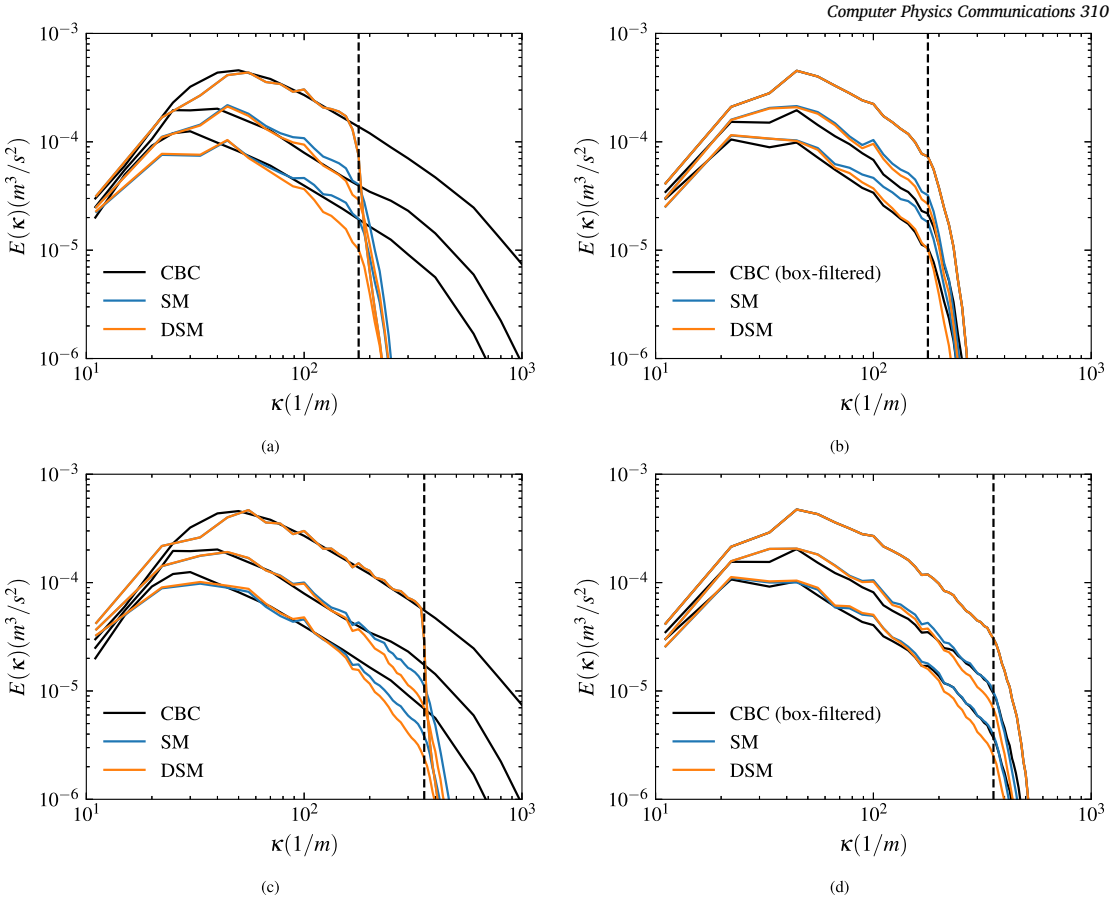


Fig. 5. Decaying isotropic turbulence: computed velocity spectra with 32^3 (a,b) and 64^3 (c,d) grid points. The experimental spectra [41] are shown unfiltered (a, c) and box-filtered (b, d). In each plot, the results for three time instants $tU_0/M = 42$, 98, and 171 are displayed from top to bottom. The vertical dashed lines denote the Nyquist limits.

to its maximum allowable value for numerical stability, multiplied by a safety factor of 0.95.

4.1. Decaying isotropic turbulence

We first validate the LES capability using freely decaying isotropic turbulence. The physical experiment was performed by Comte-Bellot and Corrsin [41], with decaying turbulence generated behind a mesh with size $M = 5.08$ cm, and freestream velocity $U_0 = 10$ m/s. The Taylor microscale Reynolds number ($Re_\lambda = u_{rms}\lambda/\nu$, where u_{rms} is the root-mean-square of a fluctuating velocity component, λ is the Taylor microscale, and ν is the kinematic viscosity) is 71.6 at $tU_0/M = 42$, decreasing to 60.6 at $tU_0/M = 171$. In a reference frame moving with the average flow velocity, the problem is modeled as freely decaying isotropic turbulence. We simulate this by considering the flow inside a cubic domain with periodic boundary conditions, where the box edge length is $9 \times 2\pi$ cm ($\approx 11M$). Two grid resolutions are employed, with 32 and 64 cells in each direction, respectively. The corresponding grid spacings are $\Delta/\eta = 60$ and 30, where η is the Kolmogorov length scale at $tU_0/M = 42$. The computations are initialized with a synthetic turbulent field whose energy spectrum matches the filtered experimental spectrum at the initial time $tU_0/M = 42$. The filtering is done either with a spectral cutoff filter or a physical box filter. When the spectral cutoff filter is applied, the initial field is directly generated on the LES grids using an open-source tool [42]. For the physical box filter, a field is first generated on a grid of 256^3, then it is filtered onto the LES grids.

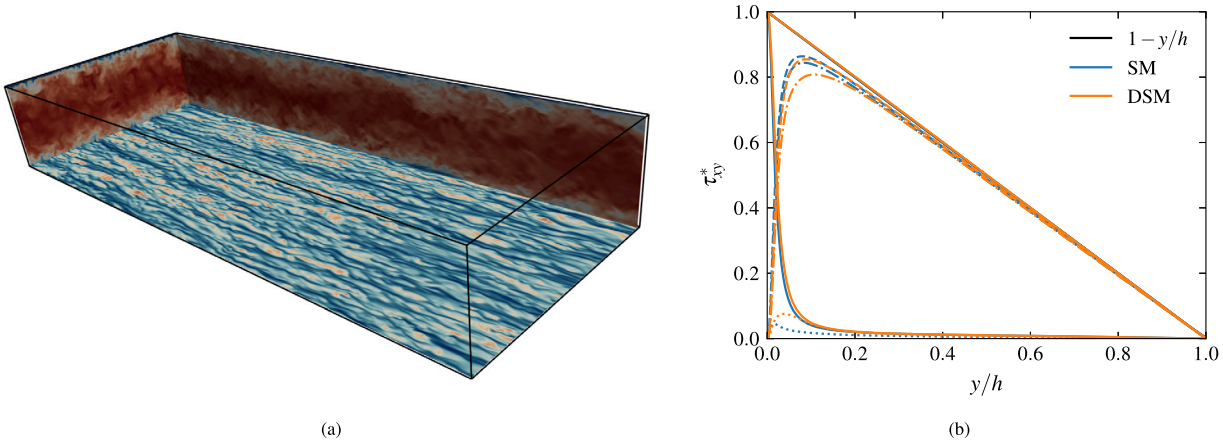
Fig. 5 compares the computed energy spectra with experimental results at three time instants, namely $tU_0/M = 42$, 98 and 171. The classical Smagorinsky model (SM) with $C_s = 0.18$ accurately predicts the results on both meshes, regardless of the filter used. The only exception is the coarse grid when box filter is applied, in which case the model does not yield sufficient subgrid-scale dissipation. Our tests show that a larger model constant, $C_s = 0.22$, yields good agreement with the filtered experimental spectra (not shown in Fig. 5). The dependence of the model constant on the grid resolution is a common drawback of static SGS models [43,44]. In contrast, the dynamic Smagorinsky model (DSM) does not rely on model constants and reasonably agrees with experimental data. Notably, the computed velocity spectra exhibit near-perfect agreement with experimental results when the box filter is applied, which aligns with the implementation of DSM, where the test filter is a box filter. Larger deviations are generally observed at the scales close to the Nyquist limits, likely due to the numerical errors associated with finite-differencing at high wavenumbers [45].

4.2. Wall-resolved turbulent plane channel flow

Fully developed channel flow is simulated in a domain of $(L_x, L_y, L_z) = (12.8h, 2.0h, 4.8h)$, with h the channel half-height. No-slip boundary conditions are imposed in the y direction and periodic boundary conditions are applied to the x and z directions. The bulk velocity is maintained constant in time using a time-varying, spatially uniform body force. The bulk Reynolds number is defined as $Re_b = 2u_b h/\nu$, with u_b the bulk velocity and ν the kinematic viscosity. The friction Reynolds number is defined as $Re_\tau = u_\tau h/\nu$, where $u_\tau = (\tau_w/\rho)^{1/2}$ is the friction velocity, and the friction coefficient is defined as $C_f = \tau_w/(\rho u_b^2/2)$. The bulk Reynolds number is 20,000 and reference DNS data [46] have $Re_{\tau,DNS} = 543.5$ and $C_{f,DNS} = 0.00591$. Table 3 reports the computational parameters for the wall-resolved LES cases, where N_x, N_y, and N_z denote the number of grid points in each direction, and Δx, Δy, and Δz represent the corresponding grid spacings. Specifically, Δy_c de-

Table 3
Computational parameters for WRLES of channel flow.

Mesh ($N_x \times N_y \times N_z$)	Δx^+	Δz^+	Δy_c^+	Δy_w^+	AR	SGS	Re_τ	C_f	ϵ_f	#ETT
$192 \times 128 \times 128$	36.2	20.4	21.5	0.59	1.8	SM	568.4	$0.00646 \pm 0.14\%$	9.38%	32.6
$288 \times 192 \times 192$	24.2	13.6	14.3	0.39	1.8	SM	567.3	$0.00644 \pm 0.17\%$	8.95%	32.6
$384 \times 256 \times 256$	18.1	10.2	10.8	0.29	1.8	SM	556.8	$0.00620 \pm 0.16\%$	4.95%	32.6
$576 \times 384 \times 384$	12.1	6.8	7.2	0.19	1.8	SM	544.8	$0.00594 \pm 0.10\%$	0.50%	32.6
$192 \times 128 \times 128$	36.2	20.4	21.5	0.59	1.8	DSM	537.3	$0.00577 \pm 0.13\%$	-2.25%	32.6
$288 \times 192 \times 192$	24.2	13.6	14.3	0.39	1.8	DSM	541.6	$0.00587 \pm 0.13\%$	-0.69%	32.6
$384 \times 256 \times 256$	18.1	10.2	10.8	0.29	1.8	DSM	539.4	$0.00582 \pm 0.16\%$	-1.49%	32.6
$576 \times 384 \times 384$	12.1	6.8	7.2	0.19	1.8	DSM	534.7	$0.00572 \pm 0.17\%$	-3.22%	32.6



(a) (b)

Fig. 6. Turbulent channel flow: visualization of streamwise velocity obtained with WRLES using the SM model on a mesh with $\Delta_z^+ = 6.8$ (a) and profiles of mean total (solid), resolved and modeled turbulent (dashed), resolved turbulent (dash-dotted), modeled turbulent (dotted), and viscous (solid) shear stress obtained with WRLES using the SM and DSM models on a grid with $\Delta_z^+ = 20.4$ (b). In (a), the wall-parallel plane located at $y^+ \approx 15$. In (b), normalization is based on the wall units of each simulation.

notes the grid spacing at the channel centerline, and Δy_w is the height of the first off-wall layer. The superscript "+" indicates normalization by DNS wall units, whereas superscript "*" used below indicates normalization by the wall units of the simulation. The symbols "SM" and "DSM" denote use of the classical Smagorinsky model with the van Driest damping function and of the dynamic Smagorinsky model, respectively. #ETT denotes the time-averaging interval, expressed in terms of the eddy turnover time $h/u_{\tau,DNS}$. Four sets of grids are used to assess grid convergence, with Δx^+ decreasing from approximately 40 to 10 while maintaining an aspect ratio of $AR = \Delta x/\Delta z = 1.8$. At the channel centerline, the wall-normal grid spacing is approximately equal to the spanwise spacing. Table 3 also includes the skin friction coefficient, with its standard uncertainty estimated using a modified batch means method [47]. Its relative error is determined as

$$\epsilon_f = \frac{C_f - C_{f,DNS}}{C_{f,DNS}}. \tag{23}$$

The results show that the SM model becomes increasingly accurate as the grid is refined, although it shows approximately 10% error on the coarsest grid. In contrast, the DSM model consistently exhibits errors of less than approximately 3% across all the grid resolutions.

Fig. 6(a) shows a visualization of the flow computed on the finest grid, where the near-wall low- and high-speed streaks are clearly observed. To verify the LES implementation, Fig. 6(b) presents the profiles of the various contributions to the total shear stress for the grid with $\Delta z^+ = 20.4$. Achievement of a linear distribution of the total shear stress provides evidence of general reliability of the LES implementation. Overall, the DSM model generates higher levels of modeled stress along with a lower peak of the resolved shear stress, as compared to the SM model with the van Driest damping function, because the DSM yields higher eddy viscosity levels.

Fig. 7 presents the mean streamwise velocity profiles obtained with the SM and DSM models. The velocity profiles normalized by the wall units of each simulation exhibit sensitivity to grid refinement, particularly with the SM model. However, although not shown, when normalized by the wall units of DNS, the velocity profiles show little dependence on grid refinement, with all four grid resolutions closely matching the DNS data. Consequently, the grid sensitivity of the velocity profiles, when normalized by the wall units of each simulation, primarily stems from the grid dependence of the skin friction, particularly when the SM model is applied (see Table 3). Fig. 8 shows the turbulent normal and shear stresses, normalized by the DNS wall units, disregarding the modeled stress. When the SM model is applied, the resolved shear stress is over-predicted, whereas it is under-predicted with the DSM model. This behavior aligns with the corresponding over- and under-estimation of the friction coefficient by the two models. As for the normal stress, the peak of the streamwise component is over-estimated, as also observed in other studies [48]. Overall, grid refinement leads to convergence of all the profiles towards the DNS data.

4.3. Wall-modeled turbulent plane channel flow

The computational setup is identical to that of the previous WRLES case, except that the bulk Reynolds number is 250,000 and that wall-model boundary conditions are imposed on the walls. The reference data [46] have $Re_{\tau,\mathrm{DNS}} = 5185.9$ and $C_{f,\mathrm{DNS}} = 0.00344$. Table 4 provides the computational parameters for the WMLES cases. The parameter definitions are the same as in Table 3, but the cell spacings are given in terms of the half-channel height. We use aspect ratios of $AR = 1.0$ and $AR = 2.0$, with thirteen grids generated through uniform refinement in all three directions for the grid convergence study. The finest grid has approximately $\Delta z^+ = 10$, which qualifies it as a WRLES case. For all the cases, the wall-modeled layer thickness is $h_{wm} = 0.1h$. Fig. 9(a) visualizes the turbulent channel flow obtained with WMLES using the SM model on a mesh with $\Delta_z/h = 0.012$. The low- and high-speed streaks are clearly visible on the plane at $y/h = 0.1$. Fig. 9(b) presents the pro-

Table 4
Computational parameters for WMLES of turbulent channel flow.

Mesh ($N_x \times N_y \times N_z$)	$\Delta x/h$	$\Delta z/h$	$\Delta y_c/h$	$\Delta y_w/h$	AR	SGS	Re_τ	C_f	ϵ_f	#ETT
$128 \times 32 \times 48$	0.100	0.100	0.100	0.0252	1.0	SM	5165.0	$0.00341 \pm 0.10\%$	-0.81%	20.7
$192 \times 48 \times 72$	0.067	0.067	0.067	0.0167	1.0	SM	5155.4	$0.00340 \pm 0.16\%$	-1.17%	20.7
$256 \times 64 \times 96$	0.050	0.050	0.050	0.0125	1.0	SM	5178.9	$0.00343 \pm 0.17\%$	-0.27%	20.7
$384 \times 96 \times 144$	0.033	0.033	0.033	0.0083	1.0	SM	5147.3	$0.00339 \pm 0.19\%$	-1.48%	20.7
$512 \times 128 \times 192$	0.025	0.025	0.025	0.0063	1.0	SM	5115.7	$0.00335 \pm 0.06\%$	-2.69%	20.7
$640 \times 160 \times 240$	0.020	0.020	0.020	0.0050	1.0	SM	5101.7	$0.00333 \pm 0.27\%$	-3.22%	20.7
$768 \times 192 \times 288$	0.017	0.017	0.017	0.0042	1.0	SM	5097.0	$0.00333 \pm 0.22\%$	-3.40%	20.7
$896 \times 224 \times 336$	0.014	0.014	0.014	0.0036	1.0	SM	5109.9	$0.00334 \pm 0.23\%$	-2.91%	20.7
$1024 \times 256 \times 384$	0.013	0.012	0.012	0.0031	1.0	SM	5121.6	$0.00336 \pm 0.24\%$	-2.46%	20.7
$1536 \times 256 \times 576$	0.008	0.008	0.012	0.0031	1.0	SM	5181.7	$0.00344 \pm 0.45\%$	-0.16%	20.7
$2048 \times 512 \times 768$	0.006	0.006	0.006	0.0016	1.0	SM	5248.3	$0.00353 \pm 0.64\%$	2.42%	20.7
$4096 \times 1024 \times 1536$	0.003	0.003	0.003	0.0008	1.0	SM	5249.3	$0.00353 \pm 0.60\%$	2.46%	20.7
$6144 \times 1536 \times 2304$	0.002	0.002	0.002	0.0005	1.0	SM	5248.4	$0.00353 \pm 0.56\%$	2.43%	20.7
$128 \times 32 \times 48$	0.100	0.100	0.100	0.0252	1.0	DSM	5229.5	$0.00350 \pm 0.27\%$	1.69%	20.7
$192 \times 48 \times 72$	0.067	0.067	0.067	0.0167	1.0	DSM	5235.3	$0.00351 \pm 0.26\%$	1.91%	20.7
$256 \times 64 \times 96$	0.050	0.050	0.050	0.0125	1.0	DSM	5243.1	$0.00352 \pm 0.08\%$	2.22%	20.7
$384 \times 96 \times 144$	0.033	0.033	0.033	0.0083	1.0	DSM	5217.6	$0.00348 \pm 0.20\%$	1.23%	20.7
$512 \times 128 \times 192$	0.025	0.025	0.025	0.0063	1.0	DSM	5181.1	$0.00344 \pm 0.09\%$	-0.19%	20.7
$640 \times 160 \times 240$	0.020	0.020	0.020	0.0050	1.0	DSM	5163.1	$0.00341 \pm 0.15\%$	-0.88%	20.7
$768 \times 192 \times 288$	0.017	0.017	0.017	0.0042	1.0	DSM	5151.3	$0.00340 \pm 0.31\%$	-1.33%	20.7
$896 \times 224 \times 336$	0.014	0.014	0.014	0.0036	1.0	DSM	5150.5	$0.00340 \pm 0.16\%$	-1.36%	20.7
$1024 \times 256 \times 384$	0.013	0.012	0.012	0.0031	1.0	DSM	5161.8	$0.00341 \pm 0.29\%$	-0.93%	20.7
$1536 \times 256 \times 576$	0.008	0.008	0.012	0.0031	1.0	DSM	5179.5	$0.00343 \pm 0.15\%$	-0.25%	20.7
$2048 \times 512 \times 768$	0.006	0.006	0.006	0.0016	1.0	DSM	5207.5	$0.00347 \pm 0.73\%$	0.83%	20.7
$4096 \times 1024 \times 1536$	0.003	0.003	0.003	0.0008	1.0	DSM	5236.4	$0.00351 \pm 0.25\%$	1.96%	20.7
$6144 \times 1536 \times 2304$	0.002	0.002	0.002	0.0005	1.0	DSM	5197.2	$0.00346 \pm 0.43\%$	0.44%	20.7
$64 \times 32 \times 48$	0.200	0.100	0.100	0.0252	2.0	SM	5227.3	$0.00350 \pm 0.22\%$	1.60%	20.7
$96 \times 48 \times 72$	0.133	0.067	0.067	0.0167	2.0	SM	5335.4	$0.00364 \pm 0.09\%$	5.85%	20.7
$128 \times 64 \times 96$	0.100	0.050	0.050	0.0125	2.0	SM	5346.2	$0.00366 \pm 0.09\%$	6.28%	20.7
$192 \times 96 \times 144$	0.067	0.033	0.033	0.0083	2.0	SM	5325.2	$0.00363 \pm 0.12\%$	5.44%	20.7
$256 \times 128 \times 192$	0.050	0.025	0.025	0.0063	2.0	SM	5264.4	$0.00355 \pm 0.09\%$	3.05%	20.7
$320 \times 160 \times 240$	0.040	0.020	0.020	0.0050	2.0	SM	5208.7	$0.00347 \pm 0.06\%$	0.88%	20.7
$384 \times 192 \times 288$	0.033	0.017	0.017	0.0042	2.0	SM	5153.5	$0.00340 \pm 0.16\%$	-1.24%	20.7
$448 \times 224 \times 336$	0.029	0.014	0.014	0.0036	2.0	SM	5113.3	$0.00335 \pm 0.18\%$	-2.78%	20.7
$512 \times 256 \times 384$	0.025	0.012	0.012	0.0031	2.0	SM	5075.8	$0.00330 \pm 0.18\%$	-4.20%	20.7
$768 \times 384 \times 576$	0.017	0.008	0.008	0.0021	2.0	SM	5057.8	$0.00327 \pm 0.37\%$	-4.88%	20.7
$1024 \times 512 \times 768$	0.013	0.006	0.006	0.0016	2.0	SM	5165.5	$0.00342 \pm 2.07\%$	-0.79%	20.7
$2048 \times 1024 \times 1536$	0.006	0.003	0.003	0.0008	2.0	SM	5196.9	$0.00346 \pm 1.12\%$	0.42%	20.7
$3072 \times 1536 \times 2304$	0.004	0.002	0.002	0.0005	2.0	SM	5192.6	$0.00345 \pm 0.53\%$	0.26%	20.7
$64 \times 32 \times 48$	0.200	0.100	0.100	0.0252	2.0	DSM	5360.1	$0.00368 \pm 0.12\%$	6.83%	20.7
$96 \times 48 \times 72$	0.133	0.067	0.067	0.0167	2.0	DSM	5356.5	$0.00367 \pm 0.08\%$	6.69%	20.7
$128 \times 64 \times 96$	0.100	0.050	0.050	0.0125	2.0	DSM	5355.5	$0.00367 \pm 0.07\%$	6.65%	20.7
$192 \times 96 \times 144$	0.067	0.033	0.033	0.0083	2.0	DSM	5338.3	$0.00365 \pm 0.08\%$	5.96%	20.7
$256 \times 128 \times 192$	0.050	0.025	0.025	0.0063	2.0	DSM	5294.1	$0.00359 \pm 0.09\%$	4.22%	20.7
$320 \times 160 \times 240$	0.040	0.020	0.020	0.0050	2.0	DSM	5244.4	$0.00352 \pm 0.04\%$	2.27%	20.7
$384 \times 192 \times 288$	0.033	0.017	0.017	0.0042	2.0	DSM	5192.8	$0.00345 \pm 0.10\%$	0.27%	20.7
$448 \times 224 \times 336$	0.029	0.014	0.014	0.0036	2.0	DSM	5142.5	$0.00339 \pm 0.13\%$	-1.67%	20.7
$512 \times 256 \times 384$	0.025	0.012	0.012	0.0031	2.0	DSM	5111.4	$0.00334 \pm 0.21\%$	-2.85%	20.7
$768 \times 384 \times 576$	0.017	0.008	0.008	0.0021	2.0	DSM	5066.6	$0.00329 \pm 0.69\%$	-4.55%	20.7
$1024 \times 512 \times 768$	0.013	0.006	0.006	0.0016	2.0	DSM	5141.1	$0.00338 \pm 0.90\%$	-1.72%	20.7
$2048 \times 1024 \times 1536$	0.006	0.003	0.003	0.0008	2.0	DSM	5205.0	$0.00347 \pm 0.72\%$	0.74%	20.7
$3072 \times 1536 \times 2304$	0.004	0.002	0.002	0.0005	2.0	DSM	5219.5	$0.00349 \pm 0.53\%$	1.30%	20.7

files of various contributions to the total shear stress for the grid with $\Delta_z/h = 0.050$. The linear distribution of total shear stress is predicted accurately, demonstrating the correct implementation of WMLES. Overall, DSM yields higher levels of modeled stress, with a lower peak in the resolved shear stress, compared to the SM model. This aligns with the WRLES results (Fig. 6(b)).

Fig. 10 presents the computed velocity results on the grids with $AR = 1.0$ and $AR = 2.0$ and Fig. 11 shows the resolved turbulent normal and shear stresses for $AR = 1.0$. Profiles on four different grids are displayed to demonstrate grid convergence. For $AR = 1.0$, the profiles agree well with the DNS data, although the resolved turbulent stress profiles show noticeable variations with grid refinement. Notably, the streamwise velocity fluctuations are over-predicted, and the resolved turbulent shear stress generally increases as the grid resolution improves. The LES-computed velocity profile in the wall-modeled layer should not be taken seriously, as it is expected to be replaced by the wall model, i.e., the logarithmic law in equation (10). In contrast, the computed velocity profiles for $AR = 2.0$ become more sensitive to grid refinement, although the results on the fine grids exhibit good accuracy.

For grid convergence study, Fig. 12 plots the variations of prediction errors with $\Delta z/h$ for different quantities, including C_f, U, $\langle uu \rangle$, $\langle vv \rangle$, $\langle ww \rangle$, and $\langle uv \rangle$. For C_f, the relative error is defined in the same way as in equation (23), and for the other quantities, the relative error is defined as

$$\epsilon_\phi = \frac{\left[\int_{y/h=0.1}^{y/h=1.0} \left(\phi - \phi_{ref}\right)^2 d\left(y/h\right)\right]^{1/2}}{\left|\int_{y/h=0.1}^{y/h=1.0} \phi_{ref}\, d\left(y/h\right)\right|}, \tag{24}$$

where ϕ denotes outer-scaled quantities. Notably, the integration is performed from $y = 0.1h$ to $y = h$, excluding the wall-modeled layer, since this layer is contaminated by significant numerical errors and is

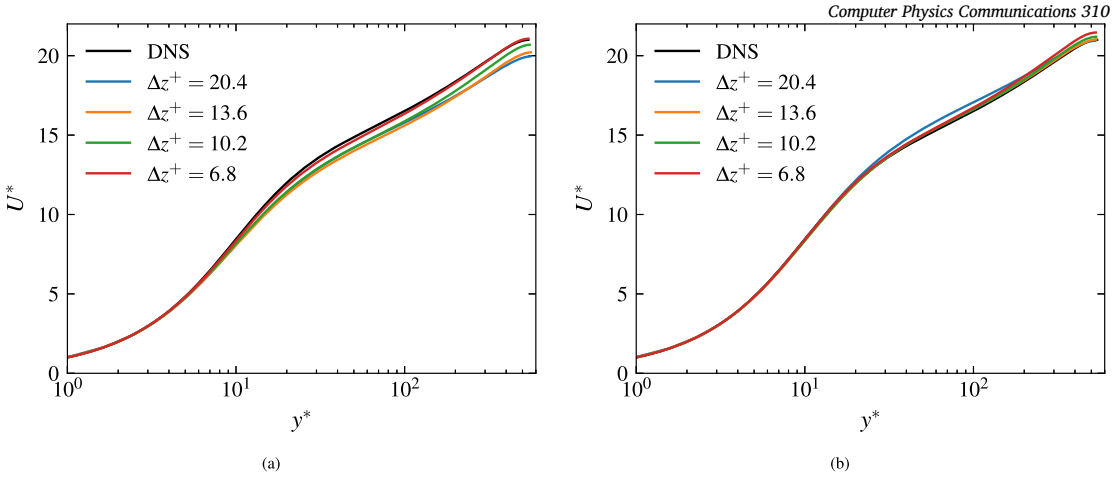


Fig. 7. Turbulent channel flow: profiles of mean streamwise velocity obtained with WRLES using the SM (a) and DSM (b) models. Normalization is based on wall units of each simulation. The DNS data is from [46].

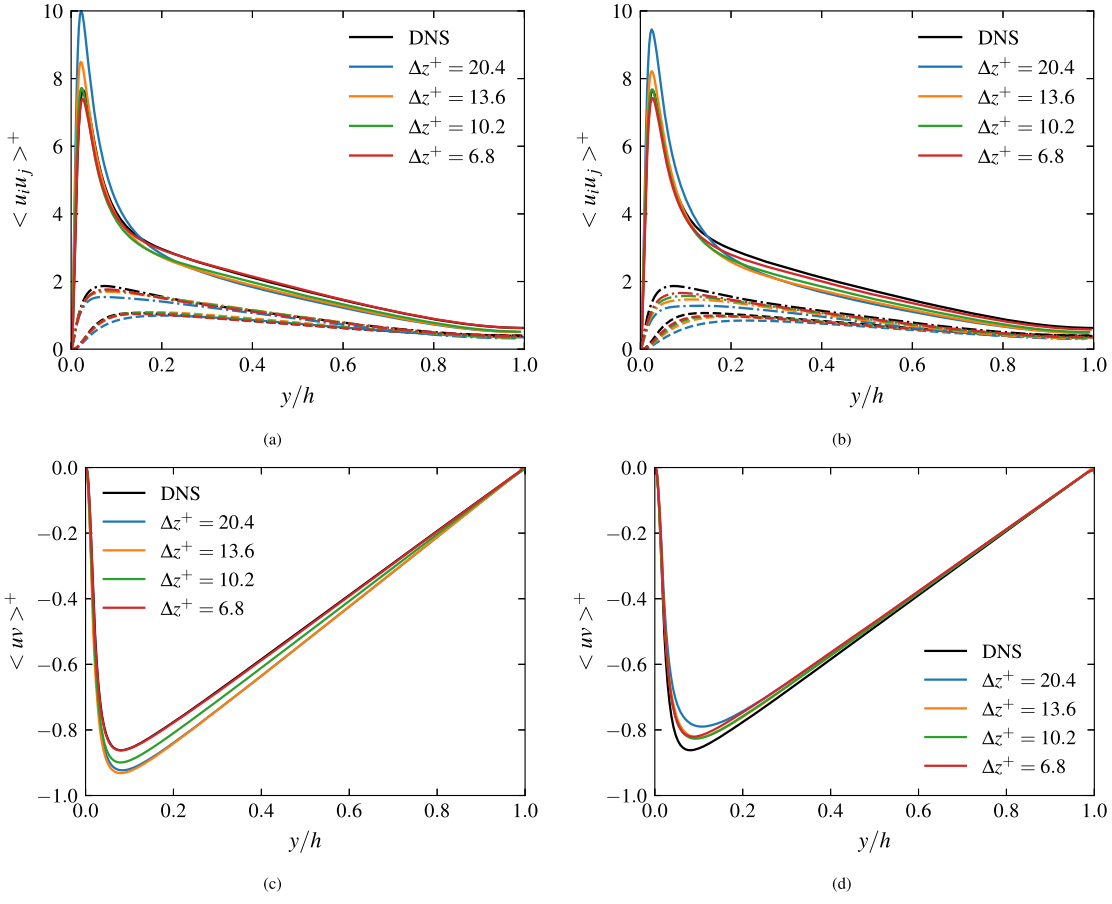
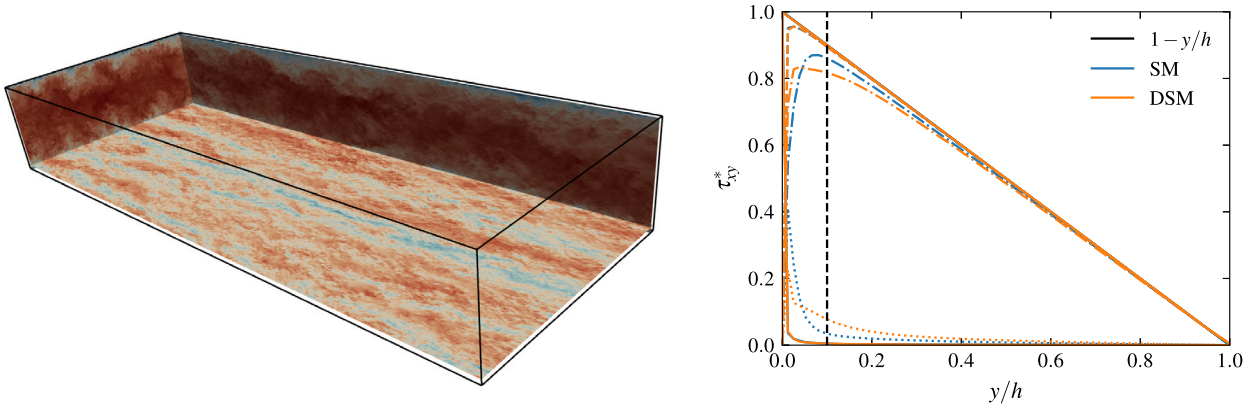


Fig. 8. Turbulent channel flow: profiles of resolved turbulent normal stress (a,b) and shear stress (c,d) obtained with WRLES using the SM (a,c) and DSM (b,d) models. Normalization is based on the DNS wall units. The DNS data is from [46]. Line codes: $\langle uu \rangle$ (solid), $\langle vv \rangle$ (dashed), $\langle ww \rangle$ (dash-dotted).

expected to be replaced by the velocity profile yielded from the wall model. Fig. 12 pinpoints non-monotonic grid convergence, for both the SGS models on grids with the two aspect ratios, as also noted by Meyers and Sagaut [49] for WRLES. In our results, the streamwise component of the turbulent normal stress exhibits the most marked non-monotonic convergence. Non-monotonic convergence is also observed in the skin friction and turbulent shear stress. On the finest grid with $AR = 1.0$, the error in wall friction is 2.43% for the SM model and 0.44% for the DSM model. The errors for the finest grid with $AR = 2.0$ are 0.26% and 1.30%, respectively. Notably, the sign of the error on the finest grids is always positive. We have also conducted WMLES for cases with $Re_b = 20,000$. Although the results are not shown here, the signs of the errors for very fine grids are also positive. This is consistent with the fact that mean wall shear stress obtained with WMLES is biased toward larger values when the instantaneous velocity is used as input for the wall model. Additionally, we note that grids with $AR = 1.0$ exhibit better grid convergence compared to those with $AR = 2.0$, as variations with grid refinement become smaller in the errors of wall shear stress and mean velocity profile, and therefore the non-monotonic grid convergence behavior becomes less pronounced.



(a) (b)

Fig. 9. Turbulent channel flow: visualization of streamwise velocity obtained with WMLES using the SM model on a mesh with $\Delta_z/h = 0.012$ (a) and profiles of mean total (solid), resolved and modeled turbulent (dashed), resolved turbulent (dash-dotted), modeled turbulent (dotted) and viscous (solid) shear stress computed with WMLES using the SM and DSM models, for the grid with $\Delta_z/h = 0.050$ and $AR = 1.0$ (b). In (a), the wall-parallel plane is located at $y/h = 0.1$. In (b), the dashed line denotes $y/h = 0.1$. Normalization is based on the wall units of each simulation.

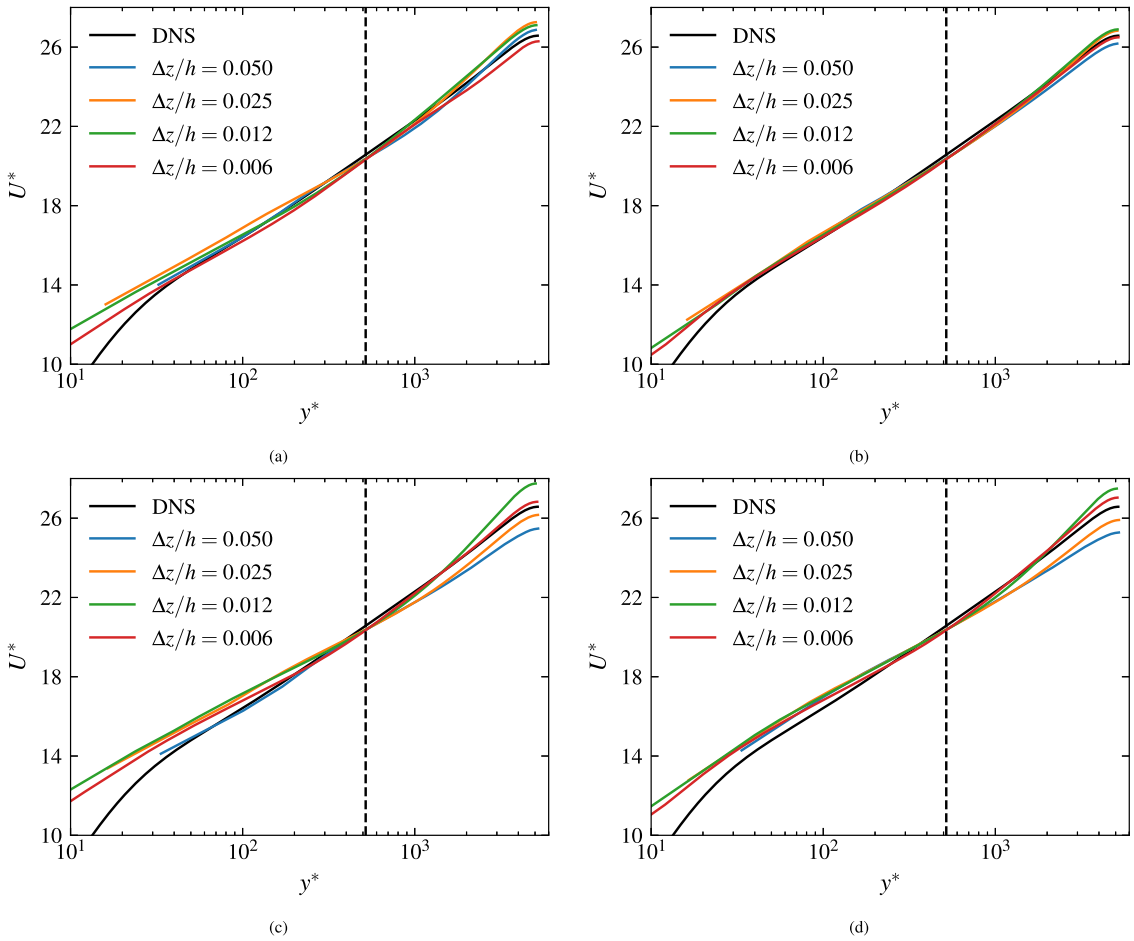


Fig. 10. Turbulent channel flow: profiles of mean streamwise velocity obtained with WMLES using the SM (a,c) and DSM (b,d) models on the grids with $AR = 1.0$ (a,b) and $AR = 2.0$ (c,d). The dashed line denotes $y/h = 0.1$. Normalization is based on the wall units of each simulation. The DNS data is from [46].

4.4. Wall-modeled turbulent square duct flow

Fully developed turbulent duct flow is simulated using WMLES, following the setup in Fig. 13(a). The computational domain is $(L_x, L_y, L_z) = (12.8h, 2.0h, 2.0h)$, where h is half of the duct side length. Wall-modeled boundary conditions are applied in the y and z directions, whereas periodic boundary conditions are enforced in the streamwise x direction. The simulation maintains the bulk velocity constant by applying a time-varying, spatially uniform body force. The bulk Reynolds number ($Re_b = 2u_b h/\nu$) is 40,000. DNS at the same Reynolds number was conducted in [50], which yielded $Re_{\tau,DNS} = 1055$, $C_{f,DNS} = 0.00557$. In our simulations, the wall-modeled layer thickness is set to $h_{wm} = 0.1h$. Table 5 provides the computational parameters, where $Re_\tau = u_\tau h/\nu$ represents the perimeter-averaged friction Reynolds num-

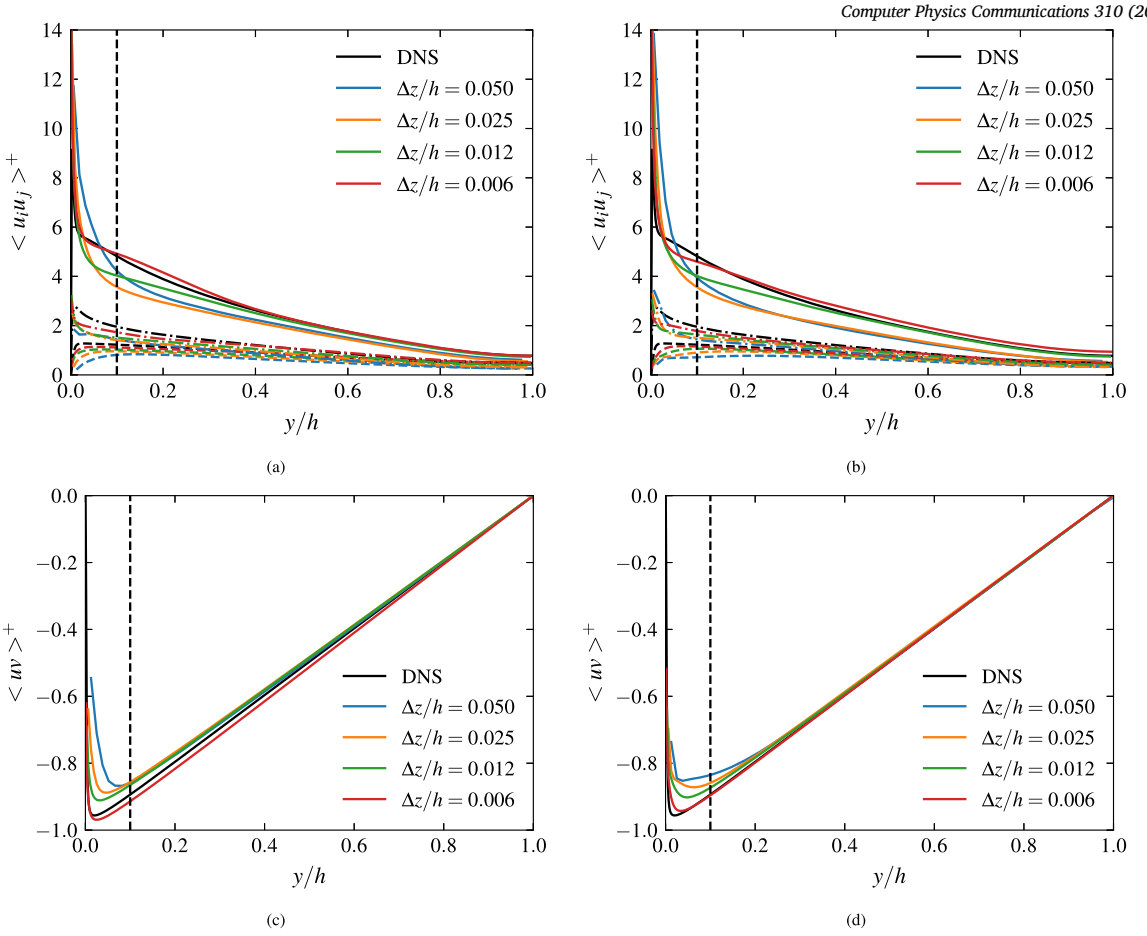


Fig. 11. Turbulent channel flow: profiles of resolved turbulent normal stress (a,b) and shear stress (c,d) obtained with WMLES using the SM (a,c) and DSM (b,d) models on the grids with $AR = 1.0$. The dashed line denotes $y/h = 0.1$. Normalization is based on the wall units of the DNS (a,b) and the wall units of each simulation (c,d). Normalization is based on the DNS wall units. The DNS data is from [46]. Line codes: $\langle uu \rangle$ (solid), $\langle vv \rangle$ (dashed), $\langle ww \rangle$ (dash-dotted).

Table 5
Parameters for the WMLES of turbulent square duct flow.

Mesh ($N_x \times N_y \times N_z$)	$\Delta x/h$	$\Delta y/h$	$\Delta z/h$	AR	SGS	Re_τ	C_f	ϵ_f	#ETT	K	ϵ_K
$128 \times 80 \times 80$	0.100	0.025	0.025	4.0	SM	1065.9	$0.00568 \pm 0.08\%$	2.08%	52.8	2.17×10^{-5}	-33.63%
$256 \times 80 \times 80$	0.050	0.025	0.025	2.0	SM	1044.5	$0.00546 \pm 0.09\%$	-1.97%	52.8	2.85×10^{-5}	-12.77%
$512 \times 80 \times 80$	0.025	0.025	0.025	1.0	SM	1038.6	$0.00539 \pm 0.11\%$	-3.08%	52.8	4.05×10^{-5}	24.05%
$768 \times 120 \times 120$	0.017	0.017	0.017	1.0	SM	1041.4	$0.00542 \pm 0.17\%$	-2.55%	52.8	4.03×10^{-5}	23.58%
$1024 \times 160 \times 160$	0.013	0.013	0.013	1.0	SM	1040.9	$0.00542 \pm 0.11\%$	-2.66%	52.8	3.67×10^{-5}	12.41%
$2048 \times 320 \times 320$	0.006	0.006	0.006	1.0	SM	1044.3	$0.00545 \pm 0.17\%$	-2.03%	52.8	3.99×10^{-5}	22.21%
$128 \times 80 \times 80$	0.100	0.025	0.025	4.0	DSM	1066.3	$0.00569 \pm 0.10\%$	2.16%	52.8	2.35×10^{-5}	-27.83%
$256 \times 80 \times 80$	0.050	0.025	0.025	2.0	DSM	1051.0	$0.00552 \pm 0.10\%$	-0.76%	52.8	3.15×10^{-5}	-3.53%
$512 \times 80 \times 80$	0.025	0.025	0.025	1.0	DSM	1046.3	$0.00547 \pm 0.10\%$	-1.65%	52.8	3.54×10^{-5}	8.37%
$768 \times 120 \times 120$	0.017	0.017	0.017	1.0	DSM	1047.0	$0.00548 \pm 0.11\%$	-1.51%	52.8	3.69×10^{-5}	13.08%
$1024 \times 160 \times 160$	0.013	0.013	0.013	1.0	DSM	1045.7	$0.00547 \pm 0.09\%$	-1.76%	52.8	3.64×10^{-5}	11.41%
$2048 \times 320 \times 320$	0.006	0.006	0.006	1.0	DSM	1044.9	$0.00546 \pm 0.13\%$	-1.91%	52.8	4.09×10^{-5}	25.29%

ber, and $C_f = \tau_w/(\rho u_b^2/2)$ is the perimeter-averaged skin friction coefficient. Six meshes with uniform grid spacings in all three directions are used for the grid convergence study, with the finest having $\Delta_z^+ = 6.3$. Fig. 13(b) visualizes the turbulent flow on the finest grid obtained with WMLES using the SM model with the van Driest damping function. This visualization clearly captures the low- and high-speed streaks, as well as the coherent structures responsible for the secondary flow. Table 5 lists the relative error in the skin friction coefficient, with absolute values within approximately 3% for all the cases. This supports the general applicability of the logarithmic law as a wall model to predict friction in flows with moderate geometrical complexity. We also report the secondary-flow kinetic energy, defined as [51]

$$K = \int_{y/h=-1}^{0} \int_{z/h=-1}^{0} \frac{1}{2} \left(\left(\frac{V}{u_b} \right)^2 + \left(\frac{W}{u_b} \right)^2 \right) d\,(y/h)\, d\,(z/h)\,, \tag{25}$$

where V and W are the mean velocity components in the y and z directions, respectively. The DNS result from [50] gives a reference value of 3.26×10^{-5}. The relative errors on the finest grid are approximately 22% for the WMLES simulation with the SM model and 25% for the DSM model. Notably, the convergence behavior of both the skin friction coefficient and the cross-flow kinetic energy is non-monotonic for both SGS models.

Fig. 14 presents contours obtained with the SM model for the mean velocity components (U^+, V^+), turbulent normal stresses ($\langle uu \rangle^+$,

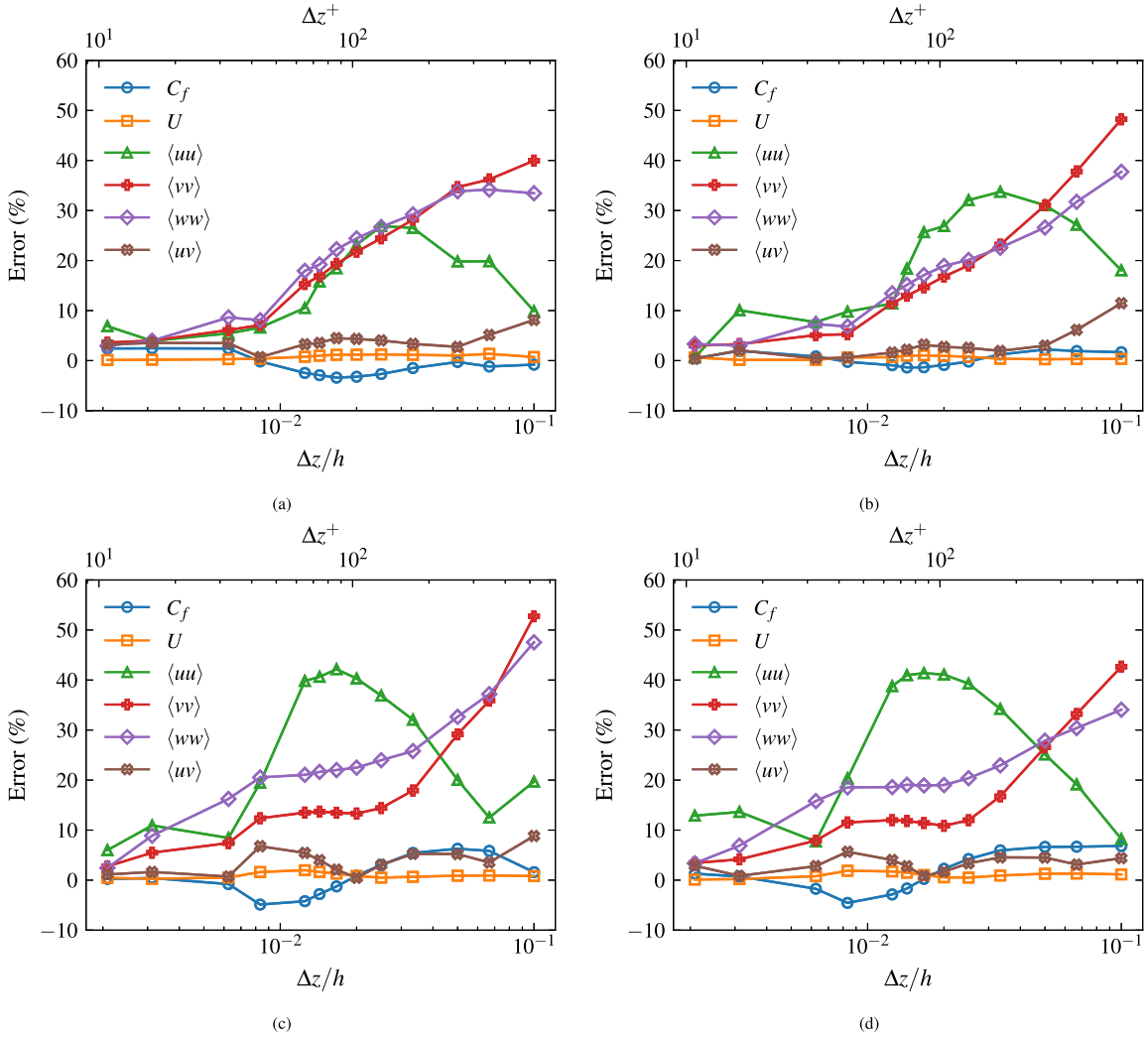


Fig. 12. Turbulent channel flow: prediction error as a function of $\Delta z/h$ or Δz^+ for various statistical properties, as obtained from WMLES with SM (a,c) and DSM (b,d) models for grids with $AR = 1.0$ (a,b) and $AR = 2.0$ (c,d).

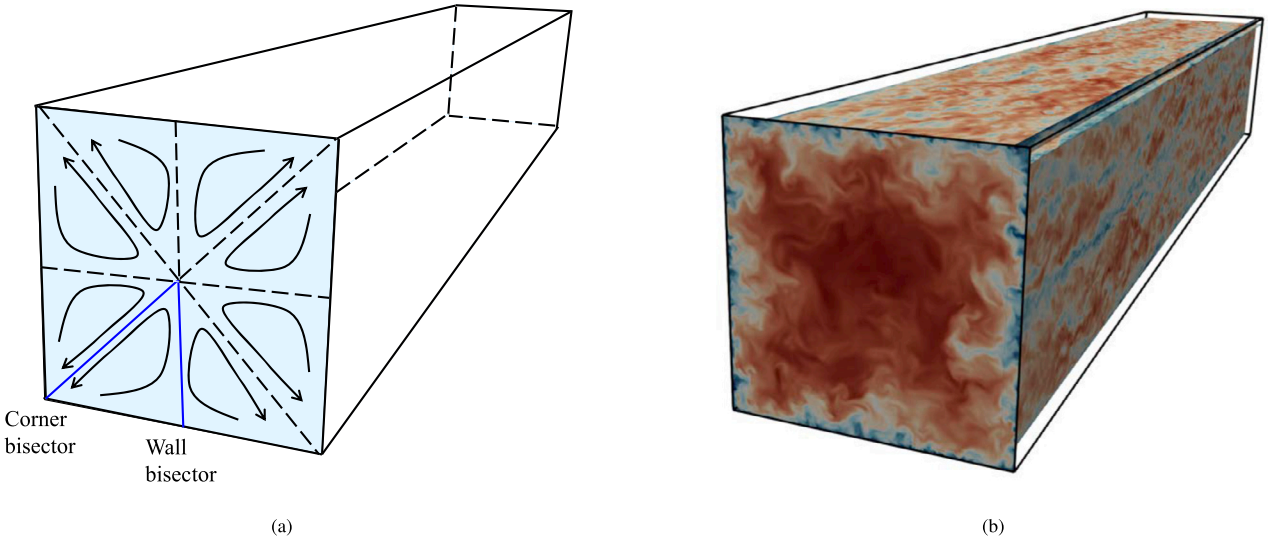


Fig. 13. Computational setup for the square duct flow (a) and visualization of turbulent flow obtained with WMLES using the SM model on a mesh with $\Delta_z/h = 0.006$ (b). The planes display contours of the streamwise velocity, with the wall-parallel plane located at $y/h = 0.1$.

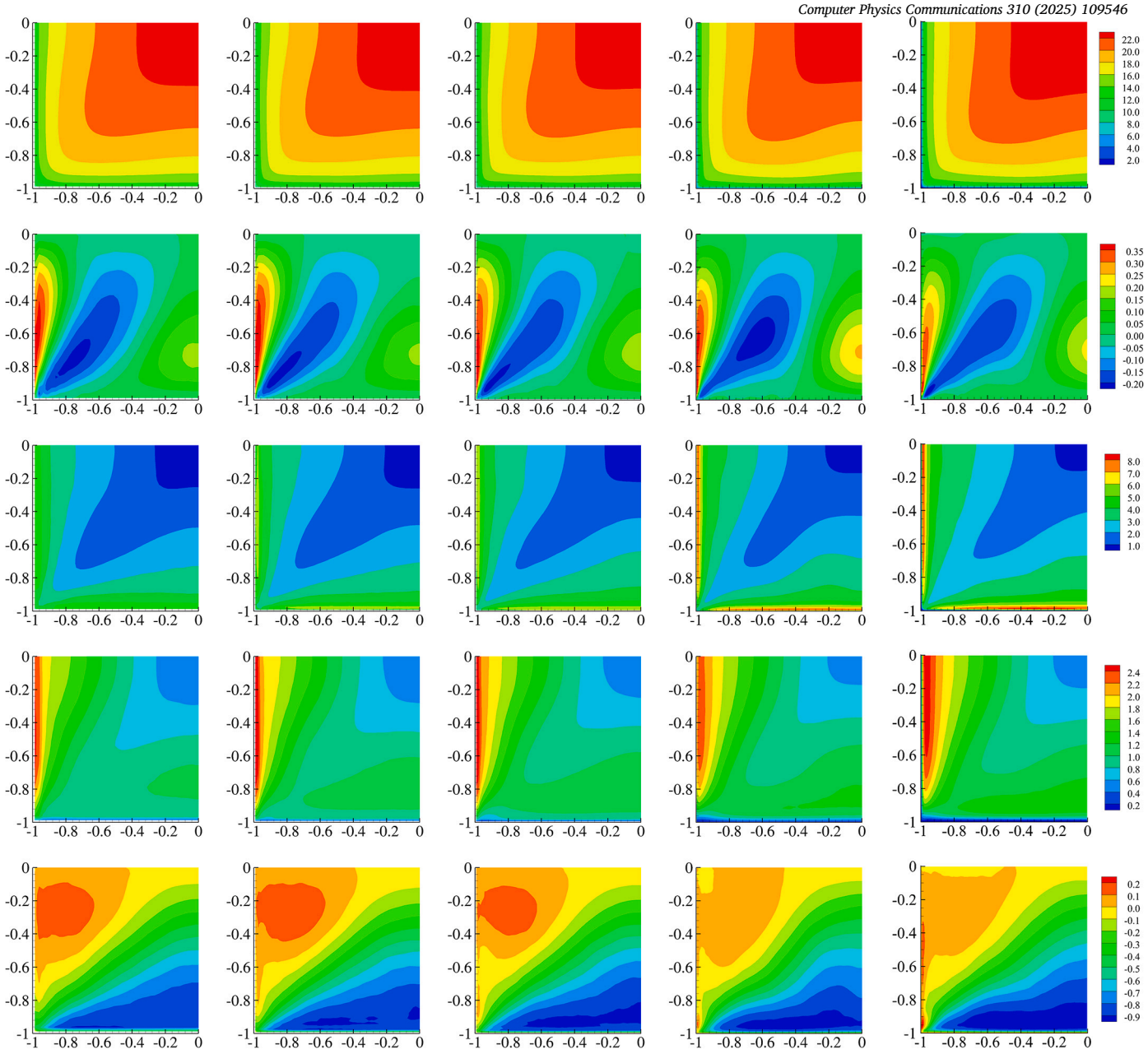


Fig. 14. Turbulent square duct flow: cross-stream contours of mean velocity U^+ (row 1) and V^+ (row 2), turbulent normal stresses $\langle uu\rangle^+$ (row 3) and $\langle vv\rangle^+$ (row 4), and shear stress $\langle uv\rangle^+$ (row 5), computed on different grids using WMLES with the SM model. Normalization is based on the DNS wall units. Only a quarter of the full domain is presented. The horizontal axis is z and the vertical axis is y. $\Delta_z/h = 0.025$ (column 1), $\Delta_z/h = 0.017$ (column 2), $\Delta_z/h = 0.013$ (column 3), $\Delta_z/h = 0.006$ (column 4) and DNS [50] (column 5).

$\langle vv\rangle^+$), and turbulent shear stress ($\langle uv\rangle^+$). Only one quarter of the full domain is presented. The contours of W^+, $\langle ww\rangle^+$, and $\langle uw\rangle^+$ are not displayed, as they are symmetric with respect to the diagonal, to those of V^+, $\langle vv\rangle^+$, and $\langle uv\rangle^+$, respectively. The distributions obtained for various grids show little variations in the core region and agree reasonably well with the reference DNS data. The primary benefit of grid refinement is an increase in the overall accuracy in the near-wall region.

We further extract the profiles of these quantities along the wall and corner bisectors (see Fig. 13(a)). Fig. 15 displays the profiles of mean streamwise velocity along both the wall bisector and the corner bisector. Consistent with the contours in Fig. 14, the mean streamwise velocity profiles show minor variations with grid refinement, indicating satisfactory grid convergence properties. In contrast, the most significant variations are observed in the turbulent normal stress $\langle uu\rangle$ (Fig. 16). Unlike the results for channel flow, the predicted peak in $\langle uu\rangle$ along the wall bisector increases with grid refinement, leading to improved accuracy. Small oscillations are observed near the turbulent stress peak for the coarser grids, which we attribute to the absence of grid clustering near the wall; these oscillations gradually disappear with grid refinement. Notably, the variations in shear stress exhibit a clearly non-monotonic pattern as the grid is refined.

5. Conclusion

We have introduced CaLES, a GPU-accelerated solver designed for large-eddy simulation of incompressible wall-bounded flows. Based on the GPU-accelerated DNS solver CaNS, CaLES demonstrates significant computational efficiency improvements through GPU parallelization using a combination of CUDA Fortran and OpenACC directives, and good scalability on massively parallel architectures. The incorporation of SGS

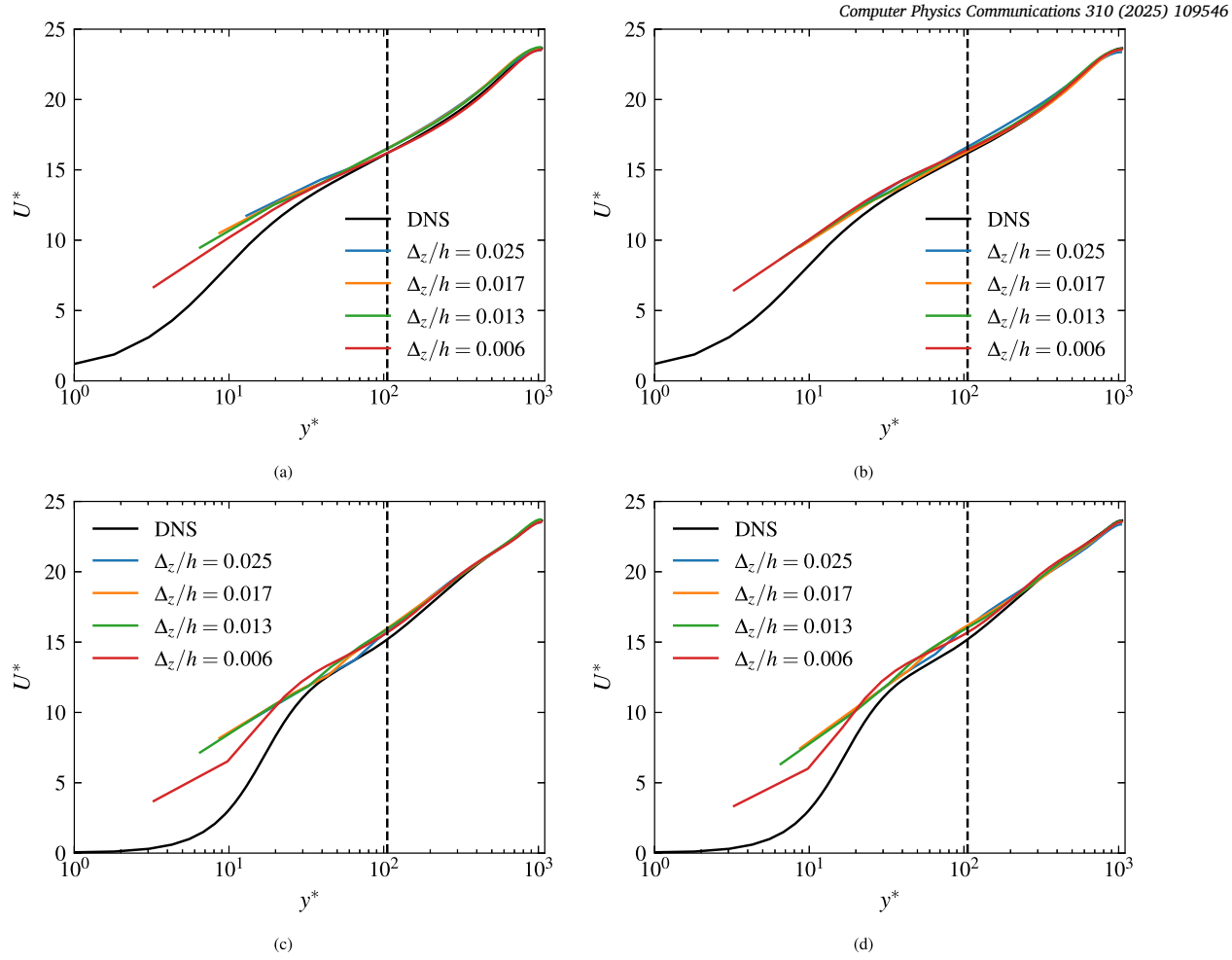


Fig. 15. Turbulent square duct flow: profiles of mean streamwise velocity along the wall bisector (a,b) and corner bisector (c,d) obtained from WMLES with the SM (a,c) and DSM (b,d) models. The dashed line denotes $y/h = 0.1$. Normalization is based on the wall units of each simulation. The DNS data is from [50].

models, including static and dynamic Smagorinsky models, along with a wall model, extends the solver's applicability to wall-resolved and wall-modeled LES.

Performance evaluation conducted on state-of-the-art high-performance computing clusters, shows that CaLES achieves substantial speed-ups using GPU acceleration compared to its CPU-only counterparts. The solver efficiently scales across multiple GPUs, achieving approximately 15× speed-up on a single GPU compared to a full CPU node, making it highly effective for high-fidelity simulations on large computational grids. Validation cases—such as decaying isotropic turbulence, turbulent channel flow, and turbulent duct flow—confirm the solver's accuracy for LES, with the dynamic Smagorinsky model exhibiting somewhat superior performance in grid convergence and turbulence prediction.

CaLES has also been applied for wall-modeled LES of turbulent channel and duct flows. Its high computational efficiency enables WMLES for channels with $Re_\tau = 5200$ on progressively refined grids, meeting the grid requirements for wall-resolved LES. A key observation is the non-monotonic grid convergence in WMLES for channel and duct flows, particularly in wall friction, streamwise velocity fluctuations, and turbulent shear stresses. These results underscore the complexities inherent in WMLES and emphasize the need for continued research into SGS and wall models.

In summary, CaLES offers the computational efficiency and flexibility needed to investigate turbulent flows with moderate complexity at high Reynolds numbers. Its open-source availability makes it a valuable tool for fast simulations, which is particularly important in the application of machine learning to develop robust SGS and wall models, including data-driven approaches and reinforcement learning.

Finally, we should emphasize that the purpose of this work is not to carry out a physical investigation of the performance of SGS models and wall models. Instead, our aim is to incorporate well-established models into a baseline solver that can easily incorporate other models, such as the Vreman subgrid-scale model [48] and the ordinary differential equation-based wall model [36].

CRediT authorship contribution statement

Maochao Xiao: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Alessandro Ceci:** Writing – review & editing, Software. **Pedro Costa:** Writing – review & editing, Software. **Johan Larsson:** Writing – review & editing, Software, Methodology, Conceptualization. **Sergio Pirozzoli:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

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

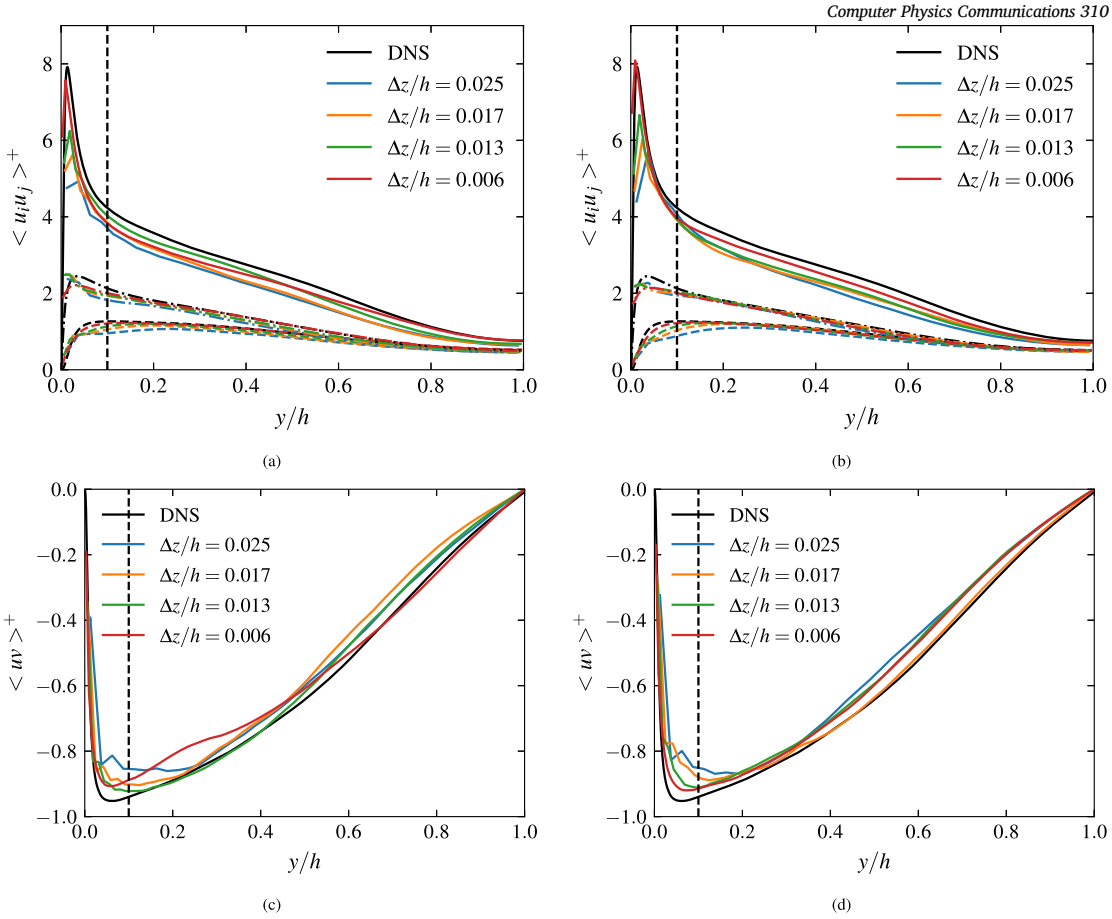


Fig. 16. Turbulent square duct flow: profiles of resolved turbulent normal stress (a,b) and shear stress (c,d) along the wall bisector as obtained from WMLES with the SM (a,c) and DSM (b,d) models. The dashed line denotes $y/h = 0.1$. Normalization is based on the DNS wall units. The DNS data is from [50]. Line codes: $\langle uu \rangle$ (solid), $\langle vv \rangle$ (dashed), $\langle ww \rangle$ (dash-dotted).

Acknowledgements

The simulations were performed using the EuroHPC Research Infrastructure resource LEONARDO, based at CINECA, Casalecchio di Reno, Italy, under a LEAP grant. Maochao Xiao would like to acknowledge the financial support provided by the China Scholarship Council, grant no. 202008610198. Special thanks are given to Di Zhou for carefully reviewing the manuscript. Additional thanks are extended to Jane H. Bae, Yufei Zhang and Xiao He for their helpful discussions.

Data availability

Data will be made available on request.

References

[1] D.C. Wilcox, Turbulence Modeling for CFD, DCW Industries, 2006.
[2] K.A. Goc, O. Lehmkuhl, G.I. Park, S.T. Bose, P. Moin, Large eddy simulation of aircraft at affordable cost: a milestone in computational fluid dynamics, Flow 1 (2021) E14.
[3] M. Xiao, Y. Zhang, F. Zhou, Numerical investigation of the unsteady flow past an iced multi-element airfoil, AIAA J. 58 (2020) 3848–3862.
[4] C.P. Arroyo, J. Dombard, F. Duchaine, L. Gicquel, B. Martin, N. Odier, G. Staffelbach, Towards the large-eddy simulation of a full engine: integration of a 360 azimuthal degrees fan, compressor and combustion chamber. Part I: methodology and initialisation, J. Global Power Propul. Soc. (2021) 133115.
[5] T. Mauery, J. Alonso, A. Cary, V. Lee, R. Malecki, D. Mavriplis, G. Medic, J. Schaefer, J. Slotnick, A guide for aircraft certification by analysis, Technical Report, 2021.
[6] J. Larsson, S. Kawai, J. Bodart, I. Bermejo-Moreno, Large eddy simulation with modeled wall-stress: recent progress and future directions, Mech. Eng. Rev. 3 (2016) 15–00418.
[7] D. Dupuy, N. Odier, C. Lapeyre, Data-driven wall modeling for turbulent separated flows, J. Comput. Phys. 487 (2023) 112173.
[8] R. Agrawal, M.P. Whitmore, K.P. Griffin, S.T. Bose, P. Moin, Non-Boussinesq subgrid-scale model with dynamic tensorial coefficients, Phys. Rev. Fluids 7 (2022) 074602.
[9] M. Gamahara, Y. Hattori, Searching for turbulence models by artificial neural network, Phys. Rev. Fluids 2 (2017) 054604.
[10] M. Kang, Y. Jeon, D. You, Neural-network-based mixed subgrid-scale model for turbulent flow, J. Fluid Mech. 962 (2023) A38.
[11] H.J. Bae, A. Lozano-Duran, Numerical and modeling error assessment of large-eddy simulation using direct-numerical-simulation-aided large-eddy simulation, arXiv preprint, arXiv:2208.02354, 2022.
[12] A. Lozano-Durán, H.J. Bae, Machine learning building-block-flow wall model for large-eddy simulation, J. Fluid Mech. 963 (2023) A35.
[13] G. Novati, H.L. de Laroussilhe, P. Koumoutsakos, Automating turbulence modelling by multi-agent reinforcement learning, Nat. Mach. Intell. 3 (2021) 87–96.
[14] H.J. Bae, P. Koumoutsakos, Scientific multi-agent reinforcement learning for wall-models of turbulent flows, Nat. Commun. 13 (2022) 1443.
[15] K. Duraisamy, Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence, Phys. Rev. Fluids 6 (2021) 050504.
[16] D. Zhou, H.J. Bae, Sensitivity analysis of wall-modeled large-eddy simulation for separated turbulent flow, J. Comput. Phys. 506 (2024) 112948.
[17] X.I. Yang, M. Abkar, G. Park, Grid convergence properties of wall-modeled large eddy simulations in the asymptotic regime, J. Fluids Eng. 146 (2024).
[18] M. Bernardini, D. Modesti, F. Salvadore, S. Sathyanarayana, G. Della Posta, S. Pirozzoli, STREAmS-2.0: supersonic turbulent accelerated Navier-Stokes solver version 2.0, Comput. Phys. Commun. 285 (2023) 108644.
[19] F. De Vanna, G. Baldan, URANOS-2.0: improved performance, enhanced portability, and model extension towards exascale computing of high-speed engineering flows, Comput. Phys. Commun. (2024) 109285.
[20] X. Zhu, E. Phillips, V. Spandan, J. Donners, G. Ruetsch, J. Romero, R. Ostilla-Mónico, Y. Yang, D. Lohse, R. Verzicco, et al., AFiD-GPU: a versatile Navier–Stokes solver for wall-bounded turbulent flows on GPU clusters, Comput. Phys. Commun. 229 (2018) 199–210.
[21] LESGO: a parallel pseudo-spectral large-eddy simulation code, https://lesgo.me.jhu.edu, 2024.

[22] P. Costa, E. Phillips, L. Brandt, M. Fatica, GPU acceleration of CaNS for massively-parallel direct numerical simulations of canonical fluid flows, Comput. Math. Appl. 81 (2021) 502–511.
[23] P. Costa, A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows, Comput. Math. Appl. 76 (2018) 1853–1862.
[24] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier-Stokes equations, J. Comput. Phys. 59 (1985) 308–323.
[25] A.A. Wray, Minimal-storage time advancement schemes for spectral methods, Technical Report MS 202, NASA Ames Research Center, 1990.
[26] F.H. Harlow, J.E. Welch, et al., Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (1965) 2182.
[27] R. Verstappen, A. Veldman, Symmetry-preserving discretization of turbulent flow, J. Comput. Phys. 187 (2003) 343–368.
[28] U. Schumann, R.A. Sweet, Fast Fourier transforms for direct solution of Poisson's equation with staggered boundary conditions, J. Comput. Phys. 75 (1988) 123–137.
[29] J. Smagorinsky, General circulation experiments with the primitive equations: I. the basic experiment, Mon. Weather Rev. 91 (1963) 99–164.
[30] E.R. Van Driest, On turbulent flow near a wall, J. Aeronaut. Sci. 23 (1956) 1007–1011.
[31] M. Germano, U. Piomelli, P. Moin, W.H. Cabot, A dynamic subgrid-scale eddy viscosity model, Phys. Fluids A, Fluid Dyn. 3 (1991) 1760–1765.
[32] D. Lilly, A proposed modification of the Germano subgrid-scale closure method, Phys. Fluids A, Fluid Dyn. 4 (1992) 633–635.
[33] D.K. Lilly, The representation of small-scale turbulence in numerical simulation experiments, in: Proc. IBM Sci. Comput. Symp. on Environmental Science, 1967, pp. 195–210.
[34] J.W. Deardorff, A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers, J. Fluid Mech. 41 (1970) 453–480.
[35] N. Nikitin, F. Nicoud, B. Wasistho, K. Squires, P.R. Spalart, An approach to wall modeling in large-eddy simulations, Phys. Fluids 12 (2000) 1629–1632.
[36] S. Kawai, J. Larsson, Wall-modeling in large eddy simulation: length scales, grid resolution, and accuracy, Phys. Fluids 24 (2012) 015105.
[37] I. Bermejo-Moreno, L. Campo, J. Larsson, J. Bodart, D. Helmer, J.K. Eaton, Confinement effects in shock wave/turbulent boundary layer interactions through wall-modelled large-eddy simulations, J. Fluid Mech. 758 (2014) 5–62.
[38] A.A. Amsden, The SMAC method: a numerical technique for calculating incompressible fluid flows, Los Alamos Sci. Lab. Rep., 1970.
[39] OpenACC programming and best practices guide, https://www.openacc.org/sites/default/files/inline-files/OpenACC_Programming_Guide_0.pdf, 2021. (Accessed 14 October 2024).
[40] J. Romero, P. Costa, M. Fatica, Distributed-memory simulations of turbulent flows on modern GPU systems using an adaptive pencil decomposition library, in: Proceedings of the Platform for Advanced Scientific Computing Conference, 2022.
[41] G. Comte-Bellot, S. Corrsin, Simple Eulerian time correlation of full- and narrow-band velocity signals in grid-generated, 'isotropic' turbulence, J. Fluid Mech. 48 (1971) 273–337.
[42] T. Saad, D. Cline, R. Stoll, J.C. Sutherland, Scalable tools for generating synthetic isotropic turbulence with arbitrary spectra, AIAA J. 55 (2017) 327–331.
[43] R. Cocle, L. Bricteux, G. Winckelmans, Scale dependence and asymptotic very high Reynolds number spectral behavior of multiscale subgrid models, Phys. Fluids 21 (2009).
[44] C. Meneveau, T.S. Lund, The dynamic Smagorinsky model and scale-dependent coefficients in the viscous range of turbulence, Phys. Fluids 9 (1997) 3932–3934.
[45] F. Nicoud, H.B. Toda, O. Cabrit, S. Bose, J. Lee, Using singular values to build a subgrid-scale model for large eddy simulations, Phys. Fluids 23 (2011) 085106.
[46] M. Lee, R.D. Moser, Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$, J. Fluid Mech. 774 (2015) 395–415.
[47] S. Russo, P. Luchini, A fast algorithm for the estimation of statistical error in DNS (or experimental) time averages, J. Comput. Phys. 347 (2017) 328–340.
[48] A. Vreman, An eddy-viscosity subgrid-scale model for turbulent shear flow: algebraic theory and applications, Phys. Fluids 16 (2004) 3670–3681.
[49] J. Meyers, P. Sagaut, Is plane-channel flow a friendly case for the testing of large-eddy simulation subgrid-scale models?, Phys. Fluids 19 (2007) 048105.
[50] S. Pirozzoli, D. Modesti, P. Orlandi, F. Grasso, Turbulence and secondary motions in square duct flow, J. Fluid Mech. 840 (2018) 631–655.
[51] R. Vinuesa, P. Schlatter, H. Nagib, Secondary flow in turbulent ducts with increasing aspect ratio, Phys. Rev. Fluids 3 (2018) 054606.