ANL-23/59

Turbulence Modeling with Nek5000/RS, SOD2D and Alya

Mathematics and Computer Science

ANL-23/59

Turbulence Modeling with Nek5000/RS, SOD2D and Alya

prepared by

Vishal Kumar[1], Oriol Lehmkuhl[2], Ananias Tombouldies[3], Paul Fischer[1,4,5], and Misun Min[1]

[1] Mathematics and Computer Science Division, Argonne National Laboratory
[2] Barcelona Supercomputing Center (BSC)
[3] Department of Mechanical Engineering, Aristotle University of Thessaloniki
[4] Department of Computer Science, University of Illinois Urbana-Champaign
[5] Mechanical Science & Engineering, University of Illinois Urbana-Champaign

September 25, 2023

Contents

Executive Summary . . . ii
1 Introduction . . . 1
2 Methodology . . . 1
  2.1 Nek5000/NekRS . . . 1
  2.2 SOD2D . . . 3
  2.3 Alya . . . 5
  2.4 Wall-resolved large-eddy simulation . . . 5
  2.5 Wall-modelled large-eddy simulations . . . 7
  2.6 Averaging operators . . . 12
3 Results . . . 12
  3.1 Benchmark case I: Periodic channel flow . . . 12
  3.2 Benchmark Case II: Periodic pipe flow . . . 19
  3.3 Implementation and testing of wall-models for LES in Nek5000 . . . 27
4 Conclusion . . . 29
Acknowledgments . . . 30
References . . . 32

Executive Summary

We present Validation and Verification (V & V) study of two high-order spectral element method (SEM) based Computational Fluid Dyanimcs (CFD) codes that we will utilize for work on turbulence modeling: Nek5000 and SOD2D. While the former solves the incompressible form of Navier-Stokes equation, the latter works with compressible set of equations and uses an entropy-viscosity formulation to account for the discontinuities for high Mach number flows. We demonstrate the accuracy of these codes for two benchmark sub-sonic turbulent flows: periodic channel and pipe flow, by carrying out first and second order statistical analysis, grid convergence and turbulent structure analysis using wall-resolved large eddy simulations (WRLES). Later, we report on the implementation and testing of various wall-modeling strategies for large eddy simulation of turbulent flows in Nek5000. These include both classical log-law based and decision-tree based machine-learning models. Accuracy of these closure strategies are analyzed and necessary future work is outlined.

1 Introduction

In spite of the advances in computing power in the recent decade a resolved study, such as direct numerical (DNS) or wall-resolved large-eddy simulation (WRLES), of a practical turbulent flow remains unfeasible [1, 2]. The focus then is to arrive at low resource demanding turbulent modeling strategies and to work on making them simple, fast and accurate by engineering standards. Wall-modeled large eddy simulation (WMLES) fits in this category. The basic idea of WMLES is to circumvent the need to refine the inner region of wall-bounded turbulent flow by using appropriate boundary condition. Encouraging results have been reported in literature by using the standard log-law based wall-models [3, 4]. However, since the formulation implicitly assumes an equilibrium in the turbulence dynamics, the accuracy of log-law based wall-modeling for many flows of practical applications such as pressure gradient boundary layers and separating flows remains sub-standard. Incorporating the non-equilibrium effects through the use of wall-models is not trivial. While some studies [5] consider only a few terms of the Navier-Stokes equations for non-equilibrium effects, others [6, 7] have pointed the need to either retain or discard all the terms. One approach to capture turbulence dynamics in wall-models is by the application of machine-learning (ML) strategies for model development. This circumvents the need to perform a term-by-term analysis of the Navier-Stokes and instead diverts the focus on model development based on available high-fidelity data-sets.

With these considerations in mind, we report on our effort to implement and test various turbulence closure strategies for WMLES. The studies are conducted using open-source Navier-Stokes (NS) solvers Nek5000/RS, developed at Argonne National Laboratory, and SOD2D, developed at Barcelona Supercomputing Center. Both of these solvers use high-order spectral element (SE) discretizations [8]. Furthermore, both SOD2D [9] and NekRS [10] are GPU-accelerated, targeting application simulations on various acceleration-device based exascale computing platforms [10, 11].

This report is organized as follows: Section 2 discusses the numerical framework used for the present work. Section 3 presents the results for V & V studies on periodic channel and pipe flows. Later, in Section 3 we also present preliminary results for the implementation of wall-models in Nek5000. Finally, we conclude our study in Section 3 by outlining the current limitations and necessary future work.

2 Methodology

2.1 Nek5000/NekRS

Nek5000 [12] is a spectral element code that is used for a wide range of thermal-fluids applications. Nek5000 has scaled to millions of MPI ranks using the Nek-based gsLib communication library to handle all near-neighbor and other stencil type communications [13]. On CPUs, tensor contractions constitute the principal computational kernel (typically $> 90\%$ of the flops). These can be cast as small dense matrix-matrix products resulting in high performance with a minor amount of tuning.

Nek5000 employs high-order spectral elements (SEs) [14] in which the solution, data, and test functions are represented as *locally structured* Nth-order tensor product polynomials on a set of E *globally unstructured* curvilinear hexahedral brick elements. The approach yields two principal benefits. First, for smooth functions such as solutions to the incompressible NS equations, high-order polynomial expansions yield exponential convergence with approximation order, implying a significant reduction in the number of unknowns ($n \approx EN^3$) required to reach engineering tolerances. Second, the locally structured forms permit local lexicographical ordering with minimal indirect addressing and, crucially, the use of tensor-product sum factorization to yield low $O(n)$ storage costs and $O(nN)$ work complexities [15]. As we demonstrate, the leading order $O(nN)$ work terms can be cast as small dense matrix-matrix products (tensor contractions) with favorable $O(N)$ work-to-storage ratios (computational intensity) [8].

2.1.1 Governing equations

The incompressible form of Navier–Stokes (NS) equations are given by,

$$\frac{\partial u_j}{\partial x_j} = 0, \tag{1}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + 2\nu \frac{\partial S_{ij}}{\partial x_j} + f_i \tag{2}$$

where u_i is the ith component of the velocity vector, ρ is the density, p is the fluid pressure, ν is the fluid kinematic viscosity and $S_{ij} = 0.5\left(^{\partial u_i}/_{\partial x_j} + ^{\partial u_j}/_{\partial x_i}\right)$ is the velocity strain tensor.

2.1.2 Spatial discretization and temporal integration

Time integration in Nek5000/RS is based on a semi-implicit splitting scheme using kth-order backward differences (BDFk) to approximate the time derivative coupled with implicit treatment of the viscous and pressure terms and kth-order extrapolation (EXTk) for the remaining advection and forcing terms. This approach leads to independent elliptic subproblems comprising a Poisson equation for the pressure, a coupled system of Helmholtz equations for the three velocity components, and an additional Helmholtz equation for the potential temperature. The pressure Poisson equation is obtained by taking the divergence of the momentum equation and forcing $\frac{\partial u_i^n}{\partial x_i} = 0$ at time $t^n = n\Delta t$. The velocity and temperature Helmholtz equations are obtained once p^n is known:

$$-\frac{\partial^2}{\partial x_j \partial x_j} p^n = q^n, \tag{3}$$

$$\frac{\beta_0}{\Delta t} u_i^n - \frac{\partial}{\partial x_j}\left(\frac{1}{Re} + \gamma \nu_t\right) 2S_{ij}^n = -\frac{\partial}{\partial x_i} p^n + r_i^n, \tag{4}$$

$$\left[\frac{\beta_0}{\Delta t} - \frac{\partial}{\partial x_j}\left(\frac{1}{Pe} + \frac{\gamma \nu_t}{Pr_t}\right)\frac{\partial}{\partial x_j}\right]\theta^n = s^n, \tag{5}$$

where β_0 is an order-unity constant associated with BDFk [16], S_{ij} is the resolved-scale strain-rate tensor as described, and q^n, r_i^n, and s^n represent the sum of the values from the previous timesteps for the contributions from BDFk and EXTk. Also included in r_i^n and s^n are eddy-diffusivity terms coming from the mean-field eddy-diffusivity mentioned above. The fully coupled system of Helmholtz equations for the three velocity components (4) is used only when the fluctuating (isotropic) part of the SGS stress tensor is modeled using a Smagorinsky (SMG) model based on the fluctuating strain rate. When this part is modeled through the use of a high-pass filter (HPF) [17] the resulting Helmholtz equations for the three velocity components are not coupled.

With the given time-splitting, we recast (3)–(5) into weak form and derive the spatial discretization by restricting the trial and test spaces to be in the finite-dimensional space spanned by the spectral element basis. The discretization leads to a sequence of symmetric positive definite linear systems for pressure, velocity, and temperature. Velocity and temperature are diagonally dominant and readily addressed with Jacobi-precondition conjugate gradient iteration. The pressure Poisson solve is treated with GMRES using p-multigrid as a preconditioner. Details of the formulation can be found in [18, 16, 19].

2.2 SOD2D

2.2.1 Physical model

The Navier-Stokes equations for compressible flow defined on a bounded domain Ω are given by

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla \cdot \boldsymbol{u} &= \mathbf{0} \quad \text{on} \quad \Omega \times (t_o, t_f), &(6)\\
\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p - \nabla \cdot \boldsymbol{\tau} &= \boldsymbol{f} \quad \text{on} \quad \Omega \times (t_o, t_f), &(7)\\
\frac{\partial E}{\partial t} + \nabla \cdot ((E+p)\boldsymbol{u}) - \nabla \cdot \boldsymbol{\tau u} - \nabla \cdot (\kappa \nabla T) &= \mathbf{0} \quad \text{on} \quad \Omega \times (t_o, t_f), &(8)
\end{aligned}$$

where ρ is the fluid density, $\boldsymbol{u}$ is the fluid velocity, p is the fluid pressure and $\boldsymbol{\tau}$ are the viscous stresses. In the above equations, (t_o, t_f) represents the temporal bounds for the continuous problem. For compressible fluid, the viscous stresses take the form:

$$\boldsymbol{\tau} = \mu(T) \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T - \frac{2}{3} \nabla \cdot \boldsymbol{u} \boldsymbol{I} \right), \tag{9}$$

where μ is the dynamic viscosity of the fluid. It is a function of temperature (T) of the fluid and we utilize Sutherland equation:

$$\mu(T) = 1.458\text{e} - 6 \left(\frac{T^{1.5}}{T + 110.4} \right). \tag{10}$$

In equation (8), $E \; (= \rho \, [0.5(\boldsymbol{u} \cdot \boldsymbol{u}) + e])$ is the total energy of the system that includes the contribution from internal energy (ρe) as well as the kinetic energy, $0.5(\rho \boldsymbol{u} \cdot \boldsymbol{u})$. We also define the following relations:

$$\begin{aligned}
e &= \frac{p}{\rho\,(\gamma - 1)}, &(11)\\
p &= \rho R T, &(12)\\
R &= C_p - C_v, &(13)\\
\gamma &= \frac{C_p}{C_v}. &(14)
\end{aligned}$$

Here equation (12) defines the equation of state (EOS) and C_p, C_v are the specific heats at constant pressure and volume respectively. The value of γ is fixed as 1.4 in this work, specifying an underlying assumption of a mono-atomic gas with constant specific heats.

We are using $x,\ y,\ z$ (or $x_1,\ x_2,\ x_3$) as the streamwise, vertical and spanwise directions, respectively. The corresponding filtered pressure and velocity fields are $u,\ v,\ w$ (or u_i in Einstein notation) and p. We will also be using subscripts t and n to indicate the velocity components tangential and normal to the airfoil surface. In most cases the physical quantities are normalized using the incoming velocity, U_o, the density ρ, and the airfoil chord, c. The under-resolved, subfilter-scale stresses are modelled using models particularly suited to unstructured grids. A brief overview of two such models used in the present study is given below.

2.2.2 Numerical model

SOD2D utilizes a spectral formulation of the Continuous Galerkin Finite Elements model (SEM) applied to the spatial terms in the Navier-Stokes system, which we couple with an Entropy Viscosity stabilization model, an operator split to convective terms, and a 4th order Runge-Kutta (RK) time integration scheme.

2.2.2.1 Entropy Viscosity stabilization The low-dissipation aspect of the proposed scheme stems from the use of a modified version of the Entropy Viscosity model proposed by [20]. The basic idea is that variable viscosity diffusive terms are introduced to the mass, momentum and total energy equations, with the viscosity being solution-dependent. This dependency is such that, when shocks and other discontinuities occur, the scheme is reduced to a 1st order upwind equivalent, while at smooth regimes it retains high order accuracy.

For each spectral element, first the numerical viscosity to be introduced is computed locally at each node as,

$$\mu_i^e = min(\beta^e, V_a^e) \tag{15}$$

where β^e and V_a^e are the upwind and entropy-viscosity at element e (and node a) respectively, with the upwind term being constant over all nodes of an element. Then, we filter viscosity defined by eqn. 15 using an elemental gaussian-convolution filter (9 points stencil) to get the final smoothed nodal viscosity. The upwind and entropy-viscosity terms in eqn. 15 are calculated as,

$$\beta^e = \frac{1}{2}||\rho^e\left(\sqrt{\boldsymbol{u}\cdot\boldsymbol{u}} + c^e\right)||_\infty, \tag{16}$$

$$V_a^e = C_e\frac{||R(\eta)_a^e||_\infty \rho_a^e h_e^2}{||\eta_b - \eta_b^{avg}||_\infty}, \tag{17}$$

where c^e are the nodal values of the speed of sound. Here, the term $R(\eta)_a^e$ is the entropy-viscosity residual at node a of an element e, for a given entropy function η. Notice that the denominator in equation (17) is a normalization term obtained by taking the L^1 norm of the difference between global nodal values of the entropy function (η_b) with its average (η_b^{avg}) over the entire discrete domain. The residual term $R(\eta)_a^e$ is calculated as

$$R(\eta)_b = M_{bc}^{-1} C_{bc} f_b, \tag{18}$$

where the entropy function flux, $f = \eta \boldsymbol{u}$, is computed from the updated flow variables. For compressible flows, it is convenient to use the physical definition of entropy itself by

$$\eta = \frac{\rho}{\gamma - 1} \ln(\frac{p}{\rho^\gamma}). \tag{19}$$

With the stabilization viscosity computed for each element, associated diffusive terms are added in discrete form to each transport equation. For example, for the mass transport this equates to adding $\left(-\frac{\mu_s^e}{\rho_e}\nabla^2\rho^e\right)$ on the LHS of equation (6). For the momentum and total energy equations, the existing diffusive terms are modified by adding the new viscosity to the physical and turbulence modeling ones, should they exist, although taking into account only the non-dilational components of the shear stress tensor. The total energy transport also includes a temperature diffusion term based on the entropy-viscosity, using the Prandtl number relation to transform artificial viscosity into artificial conductivity.

2.3 Alya

Some wall-model large-eddy simulations reported in this study are performed using Alya, a multi-physics, massively parallelized, unstructured finite-element simulation code developed at Barcelona Supercomputing Center [21, 22]. It has been widely validated in many turbulent-flow configurations: [23, 24, 25, 26, 27, 28].

The governing Navier-Stokes equations for incompressible flow (1) and (2) are discretised on a collocated unstructured grid by means of low dissipation second-order conservative schemes [29]. The same interpolation scheme for velocity (u_i) and pressure (p) is used in space. A third order Runge-Kutta explicit time discretization is used to advance the solution in time, combined with an eigenvalue-based time-step estimator [30]. A fractional time-step algorithm is used to solve the resulting system of linear equations [27]. The Poisson equation is solved using a Deflated Conjugate Gradient[31]; convergence and stopping criteria are based on the lagged algebraic residual. For exact details on the numerical framework of Alya reader is referred to [27].

2.4 Wall-resolved large-eddy simulation

Large eddy simulation (LES) employs a filtering technique to differentiate the contributions of different scales. The filtering (of ϕ^*) is represented mathematically as

$$\hat{\phi} = \int_{-\Delta_f/2}^{\Delta_f/2} \mathcal{F}\left(x - \xi; x\right)\, \phi^*(\xi)\, d\xi, \tag{20}$$

where Δ_f is the filter width. A spatially filtered resolved-scale formulation for the equations (1)-(2) can be writted as

$$\frac{\partial \hat{u}_j}{\partial x_j} = 0, \tag{21}$$

$$\frac{\partial \hat{u}_i}{\partial t} + \hat{u}_j \frac{\partial \hat{u}_i}{\partial x_j} = -\frac{1}{\hat{\rho}} \frac{\partial \hat{p}}{\partial x_i} + 2\nu \hat{S}_{ij} - \frac{\partial \hat{\tau}_{ij}}{\partial x_j} + f_i, \tag{22}$$

where the extra stress term is given by

$$\hat{\tau}_{ij} = \widehat{u_i u_j} - \hat{u}_i \hat{u}_j. \tag{23}$$

$\hat{\tau}_{ij}$ is referred to as the *sub-grid scale stresses* (SGS) and needs to be modeled from known flow quantities.

2.4.1 Vreman model

The Vreman [32] closure for $\hat{\tau}_{ij}$ is given by,

$$\hat{\tau}_{ij} = -2\nu_t \hat{S}_{ij} + \tau_{kk}\delta_{ij}/3, \tag{24}$$

where $\hat{S}_{ij} = 0.5(\partial_j \hat{u}_i + \partial_i \hat{u}_j)$ and δ_{ij} is the Kronecker delta function. They propose

$$\nu_t = c\sqrt{\frac{B_\beta}{\alpha_{ij}\alpha_{ij}}}, \tag{25}$$

and

$$\alpha_{ij} = \frac{\partial \hat{u}_j}{\partial x_i}, \ \beta_{ij} = \Delta_m^2 \alpha_{mi}\alpha_{mj}, \tag{26}$$

$$B_\beta = \beta_{11}\beta_{22} - \beta_{12}^2 + \beta_{11}\beta_{33} - \beta_{13}^2 + \beta_{22}\beta_{33} - \beta_{23}^2. \tag{27}$$

The model constant c is related to the Smagorinsky constant C_s by $c = 2.5C_s^2$. The choice of C_s, however, is not straightforward, with studies suggesting different numeric value for different flow configurations [32, 33]. One of the aims of the present study is to test the dependence of results for some benchmark turbulent flow problems on the choice of C_s.

2.4.2 High-Pass Filter model

An alternative to the SGS model is the relaxation term (RT) model of Stolz *et al.* [34]. The SGS stress terms of equation (21-22) is replaced with a damping factor acting only on the high wavenumber components of the solution, to arrive at the high-pass-filter (HPF) LES model:

$$\frac{\partial \hat{u}_j}{\partial x_j} = 0, \tag{28}$$

$$\frac{\partial \hat{u}_i}{\partial t} + \hat{u}_j \frac{\partial \hat{u}_i}{\partial x_j} = -\frac{1}{\hat{\rho}} \frac{\partial \hat{p}}{\partial x_i} + 2\nu \hat{S}_{ij} - \chi H_N * u_i + f_i, \tag{29}$$

where H_N is a high-pass filter that is applied element-by-element. The solutions in the SEM are locally represented on each element, Ω^e, $e = 1, \ldots, E$, as Nth-order tensor-product Lagrange interpolation polynomials in the reference element, $\hat{\Omega} := [-1, 1]^3$. This representation can readily be expressed as a tensor-product of Legendre polynomials, P_k (In fact, P_k are the bubble functions [35] built to satisfy the boundary conditions). We can express $u(x) = \sum_{k=0}^{N} \tilde{u}_k P_k(x)$ for the Nth-order polynomial approximation on $[-1, 1]$ where $\tilde{u}_k = \frac{1}{N+1} \sum_{j=0}^{N} u_j P_k(x_j)$. Our low-pass filter with m modes dampens the top m modes quadratically by defining

$$\hat{u}(x) = \mathcal{F}_N u(x) = \sum_{k=0}^{N} \sigma_k \tilde{u}_k P_k(x), \tag{30}$$

with $\{\sigma_k = \left(\frac{m-i}{m}\right)^2\}_{i=1}^{m}$ for $k = N - m + i$ and $\sigma_k = 1$ for $k \leq N - m$. For example, we have $\sigma_6 = (2/3)^2$, $\sigma_7 = (1/3)^2$, and $\sigma_8 = 0$ when using three modes ($m = 3$) for $N = 8$ and $\sigma_7 = (1/2)^2$ and $\sigma_8 = 0$ when using two modes ($m = 2$) for $N = 8$.

Our low-pass filter is applied globally once the computation per step is completed. Thus the filtered velocity is no longer divergence free. On the other hand, our high-pass filter is applied through the right-hand side as an additional term to ensure stability, mitigate overshoot, and hold the divergence-free condition. If F is the matrix operation that applies this one-dimensional low-pass filter, then the three-dimensional low-pass filter $\mathcal{F}_N$ on $\hat{\Omega}$ is given by the Kronecker product, $F \otimes F \otimes F$, and the high-pass filter [17, 36] is

$$\mathcal{H}_N u = u - \mathcal{F}_N u = u - (F \otimes F \otimes F) u. \tag{31}$$

The relaxation term $-\chi H_N * \hat{u}_i$ provides the necessary drain of energy out of a coarsely discretized system and, in the case of constant χ, can be interpreted as a secondary filter operation with the filter $Q_N * G$ applied every $1/(\chi \Delta t)$ time steps [37] with Δt being the stepsize of the time integration. Here we use $\chi = 5$, which drains adequate energy to stabilize the simulations; however, doubling the relaxation parameter to $\chi = 10$ does not significantly affect the results. The high-pass filter used in polynomial space only affects the last mode in the Legendre spectrum of the solution (velocity or temperature).

The high-wavenumber relaxation of the HPF model is similar to the approximate deconvolution approach (ADM) [38]. It is attractive in that it can be tailored to directly act on marginally resolved modes at the grid scale. Following [17, 36], the approach allows good prediction of transitional and turbulent flows with minimal sensitivity to model coefficients. Furthermore, the high-pass filters enable the computation of the structure function in the filtered or HPF structure-function model in all spatial directions even for inhomogeneous flows, removing the arbitrariness of special treatment of selected (e.g. wall-normal) directions.

2.5 Wall-modelled large-eddy simulations

In wall-modeled large-eddy simulations the need to resolve the inner layer of wall-turbulence is bypassed by applying a stress-based boundary condition [39]. The shear-stress to applied on the boundary is a function of flow dynamics away from the wall, which is still resolved to capture all the relevant structures. Figure 1 is the pictorial representation of the idea of wall-modeling in LES.

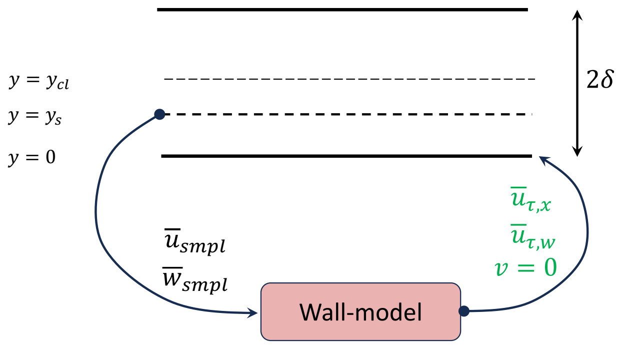

Figure 1: Illustration of the concept of a wall-modelled large-eddy simulation; y_{cl} is the channel centre-line location; y_s is the sampling location for the wall-model.

Following [40], the exact form of the traction boundary condition corresponding to the local value of the momentum flux for each of the two horizontal components of the tangential velocity $\mathbf{u}_t = \mathbf{u} - \mathbf{n}(\mathbf{n} \cdot \mathbf{u})$ and for the case where the normal to the boundary direction is aligned with the y-direction is obtained as follows:

$$\mathbf{t}_w = \tau_w \frac{\mathbf{u}_t}{|\mathbf{u}_t|}, \tag{32}$$

where

$$\tau_w = u_\tau^2. \tag{33}$$

Here τ_w to be applied as a boundary condition can be calculated from an appropriate model. These include algebraic log-law based equilibrium wall models and machine-learning based wall-models.

2.5.1 Equilibrium wall models

The simplest approach of wall-modeling is to neglect all terms in the streamwise momentum equation (7) except the Reynolds-stress gradient. This implies that the fluid acceleration, the pressure gradient and viscous effects at the sampling location are negligible. The log-law,

$$u^+ \quad = \quad \frac{|\mathbf{u}_t|}{u_\tau} = \frac{1}{\kappa}\ln(y^+) + B, \tag{34}$$

where κ is the Von-Kárman constant, B is constant of proportionality and $y^+ = {}^{y}/_{\delta_\nu}$ is the wall distance (y) in viscous length scale ($\delta_\nu = \nu/u_\tau$), governs the dynamics of wall-attached turbulent flows under the above mentioned restrictions. In this study we have taken values $\kappa = 0.41$ and $B = 5.0$, which are representative for flow over smooth walls. Two modifications to the above expression, namely Reichardt's and Spalding's wall-models, take the following form,

$$u^+ = \frac{1}{\kappa}\ln\left(1+\kappa y^+\right) + 7.8\left[1-\exp(-\frac{y^+}{11}) - \frac{y^+}{11}\exp(-\frac{y^+}{3})\right] \quad (35)$$

$$y^+ = \frac{\psi}{\kappa} + e^{-\kappa B}\left(e^{\psi} - 1 - \psi - \frac{\psi^2}{2!} - \frac{\psi^3}{3!} - \frac{\psi^4}{4!}\right) \quad (36)$$

where, using $\psi = \kappa u^+$ in eqn. 36 we arrive at 35.

2.5.2 Data driven gradient-boosted decision tree model

A brief introduction is given about the machine-learning python library used for the present work termed Extreme Gradient Boosted (XGBoost) decision tree model. The current work is based on the approach outlined in [41] for Alya. We refer to [42] for further details about XGBoost.

2.5.2.1 Introduction to decision trees ensambles For a given data set with n examples and m features $D = (x_i, y_i)(|\mathcal{D}| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, a tree ensemble model uses K additive functions to predict the output as

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i), \quad f_k \in \mathcal{F}, \quad (37)$$

where $\mathcal{F} = \left\{f(\boldsymbol{x}) = w_{q(\boldsymbol{x})}\right\}$ $(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ is the space of regression and classification trees (also termed CARTs). Here q represents the structure of each tree that maps an example to the corresponding leaf index and T is the number of leaves in the tree. Each f_k corresponds to an independent tree structure q and leaf weights w. Unlike decision trees, each regression tree contains a continuous score on each of the leaf, we use w_i to represent score on ith leaf. For a given example, we will use the decision rules in the trees (given by q) to classify it into the leaves and calculate the final prediction by summing up the score in the corresponding leaves (given by w). To learn the set of functions used in the model, we minimize the following *regularized* objective

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \quad (38)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2. \quad (39)$$

Here l is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i. The second term Ω penalizes the complexity of the model (i.e., the regression tree functions). The additional regularization term helps to smooth the final learnt weights to avoid over-fitting. Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.

2.5.2.2 Introduction to gradient boosting: XGBoost python library The tree ensemble model in (38) includes functions as parameters and cannot be optimized using traditional optimization methods in Euclidean space. Instead, the model is trained in an additive manner. Formally, let $y^(t)$

be the prediction of the ith instance at the tth iteration, we will need to add f_t to minimize the following objective

$$\mathcal{L}^{(t)} = \sum_i l\left(\hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i), y_i\right) + \Omega(f_t). \tag{40}$$

This means we add the f_t that most improves our model according to equation (38). In general, Taylor expansion of the loss function up to the second order is retained, yielding

$$\mathcal{L}^{(t)} \approx \sum_i \left[l\left(\hat{y}_i^{(t-1)}, y_i\right) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2}h_i f_t^2(\boldsymbol{x}_i)\right] + \Omega(f_t), \tag{41}$$

$$\text{where } g_i = \partial_{\hat{y}_i^{(t-1)}} l\left(\hat{y}_i^{(t-1)}, y_i\right) \text{ and } h_i = \partial^2_{\hat{y}_i^{(t-1)}} l\left(\hat{y}_i^{(t-1)}, y_i\right). \tag{42}$$

After removal of all the constants, the specific objective at step t becomes:

$$\mathcal{L}^{(t)} = \sum_i \left[g_i f_t(\boldsymbol{x}_i) + \frac{1}{2}h_i f_t^2(\boldsymbol{x}_i)\right] + \Omega(f_t), \tag{43}$$

This becomes the optimization goal for the new tree. One important advantage of this definition is that the value of the objective function only depends on g_i and h_i, such that different loss functions can be optimized using exactly the same solver that takes g_i and h_i as input.

2.5.2.3 Integration with Alya and Nek5000 A few challenges remain in order to couple external machine-learning libraries to efficiently parallelized CFD codes like Alya and Nek5000. In order to maintain performance, model training for machine learning algorithm cannot be done every time for a new simulation run. We save the model trained on any data-set and use a coupling between Alya and Python without much performance degradation. Python objects hierarchy is converted into a byte stream so that it can be reused like an algebraic model. This coupling, however, is not direct. First, the simulation data is moved to Python through C. This is done in two stages:

1. C-Foreign Function Interface (CFFI): This python library is used *(a)* to convert C datatypes to python datatypes, and *(b)* to 'embed' the python functions to fortran as a 'dynamic-link library' (this is needed because python is not an embedded programming language).
2. allow conversion of fortran datatypes to C datatypes in the main program using ISO_C_BINDING (an intrinsic library for C interoperability).

Then we compile the fortran program linking the embedded file.

2.5.3 A note on sampling and averaging

The strategy of sampling and the averaging to be performed on the sampled flow velocities (see Figure 1) are different in Nek5000 to that of SOD2D (and Alya). Utilizing the periodicity of channel flow, in Nek5000, we first perform spatial averaging of the velocity field in the homogeneous directions (streamwise and spanwise). Then, we interpolate the space averaged fields on a wall-normal line,

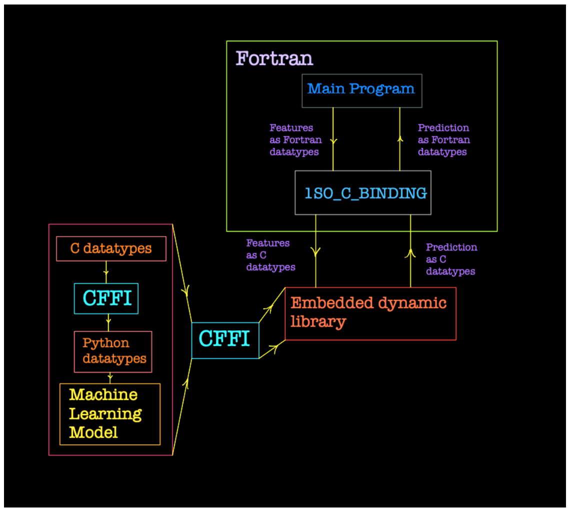

Figure 2: Schematic showing the coupling of communication between Alya's source code (in Fortran) and Python's XGBoost library.

initialized during the pre-processing stage of the simulation, at every time-step. This interpolating line has 200 data-points from $0 \le y/\delta \le 1$, stretched by a hyperbolic tangent relation as

$$\frac{y_j}{\delta} = 1 + \frac{tanh\left(\left(\frac{y_j^u}{\delta} - 1\right)\beta_m\right)}{tanh(\beta_m)}, \qquad (j = 0 \cdots N_y), \tag{44}$$

where y_j^u is the wall-normal coordinate for uniformly distributed grid, N_y is total number of grids in y-direction and β_m is a stretching parameter (we use $\beta_m = 2.0$). It should be noted that this kind of interpolation is used only for convenience as it was available for the collection of statistics in the channel. Interpolated velocity at the chosen sampling location ($y = y_s$ in Figure 1) is finally fed into the wall-model to calculate the wall-shear to be applied as a boundary condition.

In Nek5000, for the channel flow case that has two homogeneous (periodic) directions, WMLES we use the mean field eddy viscosity approach of [43] as implemented in [44], in which the SGS stress tensors τ_{ij}^{sgs} and $\tau_{\theta j}^{sgs}$ are expressed in terms of a non-isotropic, mean-field eddy viscosity (MFEV) obtained by the horizontally-averaged mean strain rate, and an isotropic, fluctuating part. The law of the wall is effected through the use of the mean-field eddy viscosity concept and the approach originally used by [40] is used to convert the horizontally-averaged traction to local values based on the local slip velocity in each of the horizontal directions. For other cases, e.g. with only one periodic direction, spatial averaging is performed only in that homogeneous direction and MFEV is not used. For more complex cases with no homogeneous directions, local instantaneous values of the velocity are sampled along the slip-wall boundary for the evaluations of wall-shear. However, this is part of an on-going investigation and is not reported here.

In SOD2D and Alya, on the other hand, we perform only short-window time averaging of the instantaneous flow velocities to be utilized in the wall-model. Furthermore, the choice of sampling location is local; each boundary node samples the velocity at a distance that is (pre) determined by the size of local mesh size. Thus, each boundary node has a unique value of wall-shear that it applies as a boundary condition.

The strategy of Nek5000 is simple and fast, such that the sampling location is independent of underlying mesh used. We observed that the wall-modeling strategy in Nek5000 also shows a faster rate of convergence of results. It is, however, suited only to periodic box flow configuration. To make it applicable to a variety of flow problems we will be working on generalizing its sampling strategy, similar to SOD2D and Alya.

2.6 Averaging operators

Finally, we note the averaging approach employed in our CFD codes. In studies of turbulent flows it is customary to average the turbulent quantities in both time and the statistically homogeneous spatial direction(s). Since we will be presenting results both for incompressible and compressible flows, it is convenient to introduce two averaging operators appropriate for such flow and establish a connection between them, with the sole purpose of comparing different flow quantities between the two codes.

For incompressible flow, we follow Reynolds decomposition of a fluctuating quantity ϕ as

$$\phi = \overline{\phi} + \phi^{'}, \tag{45}$$

where $(\bar{\cdot})$ suggests time averaging, with the property that $\overline{\phi'} = 0$. If the problem has homogeneous spatial directions, averaging will also be performed in those directions to converge to a steady state solution faster. Spatial averaging in homogeneous directions will be denoted by $\langle\cdot\rangle$. Naturally, any quantity ϕ averaged both in time-and-space will be represented by either $\langle\bar{\cdot}\rangle$ or a capital letter Φ. Finally, the turbulent stress tensor in an incompressible regime is denoted by $\overline{u_i^{'}u_j^{'}}$.

For compressible flow, on the other hand, we follow the method of density-averaged decomposition of Reynolds [45], also termed Favre averaging [46]. It is represented by

$$\phi = \widetilde{\phi} + \phi^{''}, \tag{46}$$

where $\widetilde{\phi} = \overline{\rho\phi}/\overline{\rho}$. The Favre-averaged decomposition has the property that: (1) $\overline{\widetilde{\phi}} = \widetilde{\phi}$, (2) $\widetilde{\phi''} = 0$, but (3) $\overline{\phi''} \neq 0$. Following Favre averaging simplifies the flow quantities that can be compared with those of the incompressible flow in a straightforward manner. For example, in compressible flow the turbulent stress tensor is represented as $\widetilde{u_i^{''}u_j^{''}}$. In the subsonic regime ($M_o < 0.2$), we will be referring to both $\overline{u_i^{'}u_j^{'}}$ and $\widetilde{u_i^{''}u_j^{''}}$ as "Reynolds stresses".

3 Results

3.1 Benchmark case I: Periodic channel flow

We now report on the wall-resolved large-eddy simulations of turbulent channel flow using Nek5000 and SOD2D. We will be using the data from [47], who performed DNS at the same flow configuration, to conduct validation for the results obtained.

3.1.1 Simulation setup

The simulations were conducted in a box of domain $\Omega \equiv L_x \times L_y \times L_z = 2\pi\delta \times 2\delta \times \pi\delta$, at bulk-flow Reynolds number, $Re_b = U_b\delta/\nu$, which corresponds to friction Reynolds number, $Re_\tau = u_\tau\delta/\nu = 950$. Impenetrable-wall boundary conditions are imposed on the boundaries in wall-normal direction (y) on velocity. Periodic boundary condition is utilized both in the streamwise (x) and spanwise (z) directions. For initial condition, we utilize a velocity profile given by Reichardt's log-law (35), superimposed with random fluctuations to achieve a turbulent flow. Flow forcing strategy in the two codes are somewhat different: while Nek5000 uses instantaneous flow forcing to maintain $U_b = 1$ in the streamwise direction, SOD2D utilized a steady forcing based on *apriori* knowledge of the case to achieve a unit bulk velocity. We have tested both the strategies (instantaneous *vs* steady) to drive the flow in Nek5000; former strategy results in a reduced time to mitigate the transients of the flow. The effect on final statistics, however, were found to be minimally impacted.

After initializing the simulations, flow statistics for a minimum of 100 & 25 convective time units ($1\,CTU = \delta/U_b$) were discarded for Nek5000 and SOD2D. This corresponds to approximately 15 & 5 domain flow through time for Nek5000 and SOD2D, respectively. Then statistics were collected for a minimum of 500 CTU for Nek5000 & 100 CTU for SOD2D, for the results discussed here. Together with temporal averaging, we have also averaged the flow in two homogeneous directions.

Table 1: Grid resolutions used for convergence study for the periodic channel flow at $Re_\tau = 950$; N_{elm} refers to the number of spectral elements used; N_p refers to the polynomial order used inside the spectral elements; N_{dof} is the total number of degrees of freedom (in million, M); r_s is the geometric stretching ratio employed for generating the mesh in wall-normal direction; $\Delta x^+, \Delta y^+, \Delta z^+$ denote spacing in viscous units; ϵ_{u_τ} is the error in u_τ/U_b, given by equation (47); Δt is the average simulation time-step.

	Vreman SGS model; $Re_\tau = 950$								
Case	N_{elm}	N_p	$N_{dof}(M)$	r_s	Δx^+	Δy^+	Δz^+	ϵ_{u_τ}	Δt
Nek5000	$16 \times 16 \times 16$	6	0.9	1.66	62	$(1.8 - 64.1)$	31.1	-1.9	6.6×10^{-3}
	$24 \times 24 \times 24$		2.9	1.38	40	$(1.2 - 45)$	20	0.4	5.1×10^{-3}
	$32 \times 32 \times 32$		7.1	1.28	31	$(0.87 - 35.3)$	15.5	0.7	4.6×10^{-3}
	$64 \times 64 \times 64$		56.6	1.125	15	$(0.46 - 18.0)$	7.8	1.45	2.4×10^{-3}
SOD2D	$16 \times 16 \times 16$	6	0.9	1.66	60.8	$(1.8 - 61.8)$	30	-4.1	1.2×10^{-4}
	$24 \times 24 \times 24$		2.9	1.38	41.4	$(1.28 - 44.5)$	20.7	-0.1	9.4×10^{-5}
	$32 \times 32 \times 32$		7.1	1.28	30.4	$(0.85 - 34.5)$	15.2	1.7	7.8×10^{-5}

3.1.2 Results: Grid convergence

A grid convergence study was conducted on four different resolutions: Coarse, Medium, Fine and Extra-Fine for both Nek5000 and SOD2D. The meshes were generated using the open-source grid-generation tool Gmsh[48]. Figure 3 shows the different grids used for the convergence study. We conducted a h-refinement study, whereby the elements themselves are refined for each successive refinement level, keeping the polynomial order the same. While uniform resolution was used in x

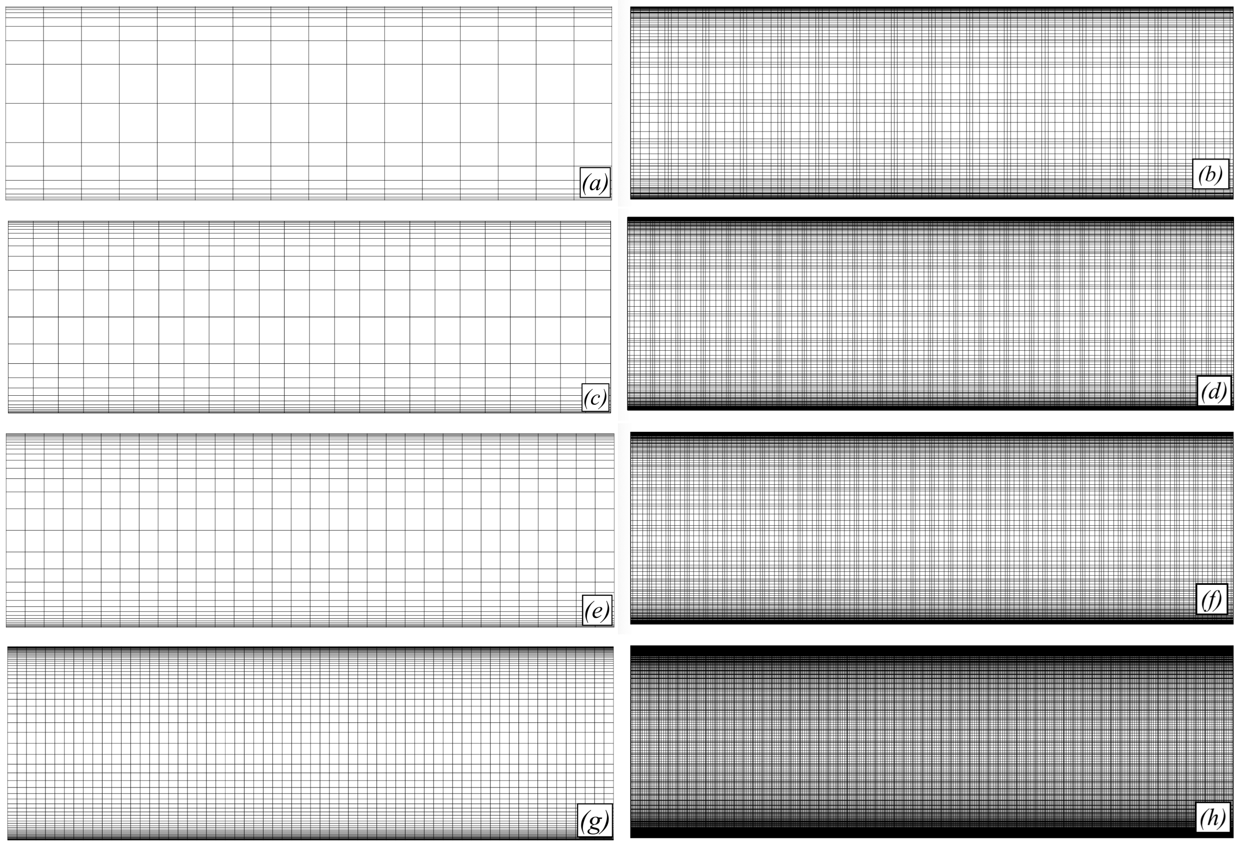

Figure 3: Longitudinal section of spectral element meshes used for turbulent channel flow simulations with both the codes; *(a,b)* Coarse mesh; *(c,d)* Medium mesh; *(e,f)* Fine mesh and *(g,h)* Extra-Fine mesh. The left column shows element boundaries while the right column shows the Gauss-Lobatto-Legendre (GLL) collocation points for a polynomial order of $N = 6$. For resolution details refer to Table 1.

and z directions, grids in wall-normal direction were stretched using geometric progression series. The stretching ratio (r_s) together with other parameters related to the resolution are reported in Table 1. We note that the Fine grid has a typical resolution ($\Delta x^+ \approx 30; \Delta y^+_{min} < 1; \Delta z^+ \approx 15$) for a wall-resolved LES calculation as mandated by various different studies in literature (see [1, 2]). Apart from these three resolutions, we also used an over-resolved mesh (termed Extra-Fine) to perform simulations with NekRS, further matching the resolutions reported by [47] shown in Figure 3*(g,h)*. However, for SOD2D the simulation on this Extra-Fine resolution becomes quite expensive due to the acoustic-based CFL condition, yielding much smaller time-steps than Nek5000. Hence, a converged statistics could not be obtained for inclusion in this report and those calculations would be reserved for future work.

Figures 4–5 show the statistical results obtained for the grid refinement study for NekRS and SOD2D, respectively. We notice that the mean streamwise velocity (U/U_b) collapses nicely with the data of [47], both in the outer (subplot *(a)*) and inner (subplot *(b)*) units, for Medium and Fine grids. Wall-distance in inner scale (or plus units) is defined as $y^+ = u_\tau \delta/\nu$, where $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity, which can be calculated from wall-shear stress ($\tau_w = \nu \partial U/\partial y|_{\text{wall}}$). Table 1 also shows the error in u_τ defined as

$$\epsilon_{u_\tau} = 100(u_\tau^* - u_\tau^{ref^*})/u_\tau^{ref^*}, \tag{47}$$

where $u_\tau^* = u_\tau/U_b$ is the non-dimensionalized friction velocity and $u_\tau^{ref^*}$ is the reference value of friction velocity taken from the data of [47]. Both Medium and Fine grid have less than 2% error in scaled friction velocity.

Figures 4 and 5*(c,d)* show the resolved root-mean-square (rms) value of normal Reynolds fluctuations $(u, v, w)_{rms}$, and resolved Reynolds-shear stress (RSS) $\left(\overline{u'v'}\right)$. They are defined as

$$u_{rms} = \sqrt{\overline{u'u'}};\ v_{rms} = \sqrt{\overline{v'v'}};\ w_{rms} = \sqrt{\overline{w'w'}}. \tag{48}$$

Overall, the corroboration between our results and the DNS data is quite good for Medium and Fine grids. Henceforth, unless otherwise stated, all the results are discussed for the Fine grid resolution.

3.1.3 Flow visualization

Figure 6 shows the instantaneous streamwise velocity for the three mesh resolutions used for the two CFD codes. We notice that refinement of the resolution results in visibly detailed capturing of instantaneous structures for both the codes. There are, however, some observable difference between the results for Nek5000 and SOD2D. For Nek5000 we can clearly see some imprint of the mesh on structures at the lowest mesh resolution (Figure 6*(a)*). Such imprints are an artifact of the C^0 continuity of the numerical method and have been reported before, most recently by [49]. However, we also clearly observe that further refinement of the mesh remove these imprints, which has also been observed in the study of [49]. Although SOD2D is also based on continuous spectral element method, the mesh imprints are suppressed in the results shown in Figure 6*(b,d,f)*.

Figure 7 shows the distribution of instantaneous entropy-viscosity for SOD2D. As expected, for low-Mach number (current study uses free-stream Mach, $M_\infty = 0.2$) flow the values remain small, with maximum scaled instantaneous values of entropy-viscosity $\nu_e \sim \mathcal{O}(10^{-2})$.

Lastly, we discuss and compare the instantaneous structures between the two codes in terms of low-speed streak spacing. Low-speed streaks are precursors to turbulence generation mechanism as outlined by various different studies (see [50, 51] among other), and an accurate prediction of such structures are fundamental to capturing the dynamics of turbulent flows. Figure 8 shows the contours of streamwise fluctuations at three different y^+ locations, representative of the distinct regions of channel flow based on flow physics: the inner region of maximum TKE production ($y^+ = 11$), the buffer region ($y^+ = 30$) and the log-law region ($y^+ = 60$) for Nek5000 (Figure 8*(a,c,e)*) and SOD2D (Figure 8*(b,d,f)*). We observe the streaky-ness of the velocity profiles quite clearly at all the locations for both the codes. We also note that as we move away from the wall the streaks becomes wider, a well-know observation made by several other studies [52]. These characteristics of wall-streaks between Nek5000 and SOD2D looks quite similar and, thus, the resolution seems adequate for capturing the turbulence dynamics.

Periodic channel flow at $Re_b = 19000$

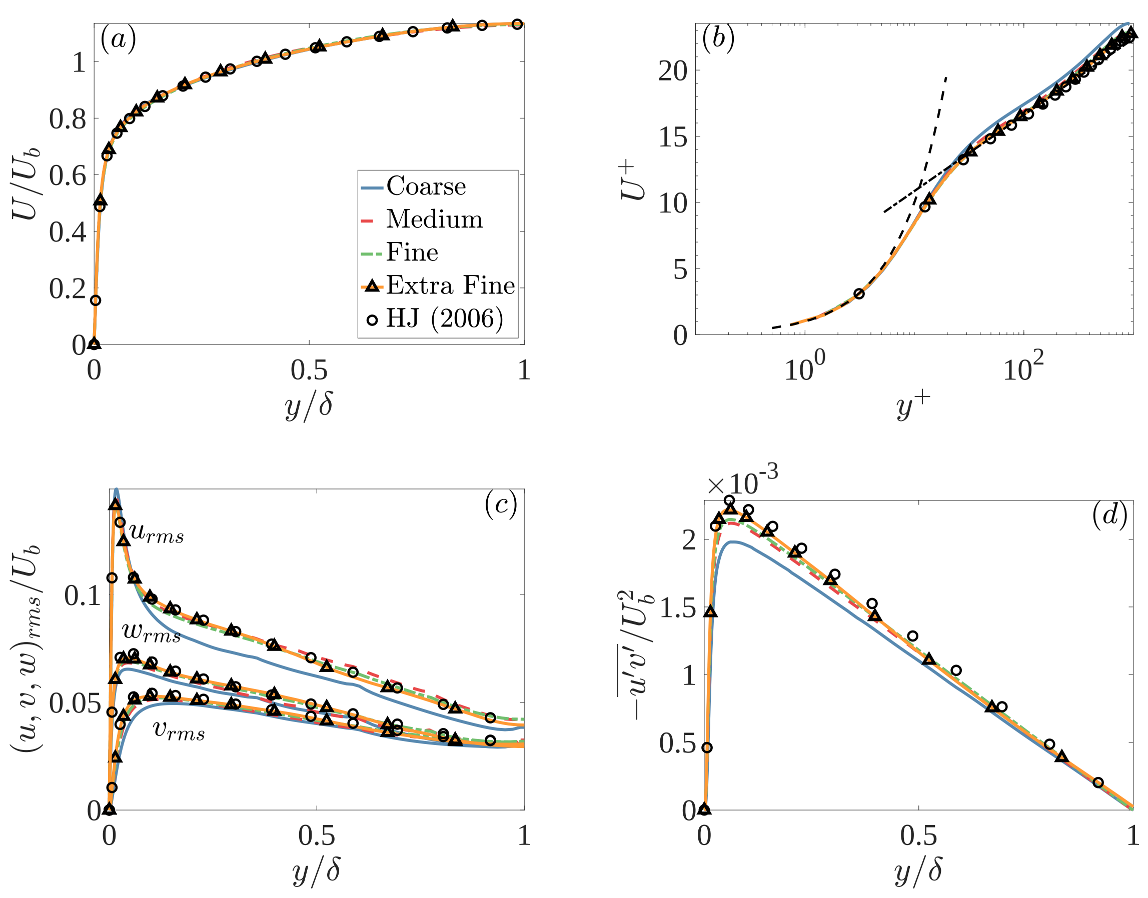

Figure 4: Grid convergence study for channel flow with NekRS at $Re_\tau = 950$; *(a)* streamwise velocity in outer units; *(b)* streamwise velocity in inner units; *(c)* RMS of normal-component of turbulent stresses; *(d)* resolved Reynolds stresses.

3.1.4 Results: Flow Statistics

Figure 9 compares the results of Nek5000, NekRS and SOD2D against each other at the Fine grid resolution mentioned in Table 1. As mentioned previously, since Nek5000 and NekRS use the same numerical framework, they yield virtually identical results. Henceforth, we will show instantaneous and statistical results only from NekRS and any conclusions drawn should also be equally applicable to Nek5000. The results for SOD2D also compare very well with that of Nek5000/RS for all the statistical quantities tested. We do, however, note a slight over-prediction of $\overline{u'v'}$ for SOD2D when compared to Nek5000/RS. This could be due to smaller time-averaging period for SOD2D as compared to Nek5000/RS mentioned in the previous section. However, since the differences are small we find both Nek5000 and SOD2D well benchmarked against each other for channel flow case.

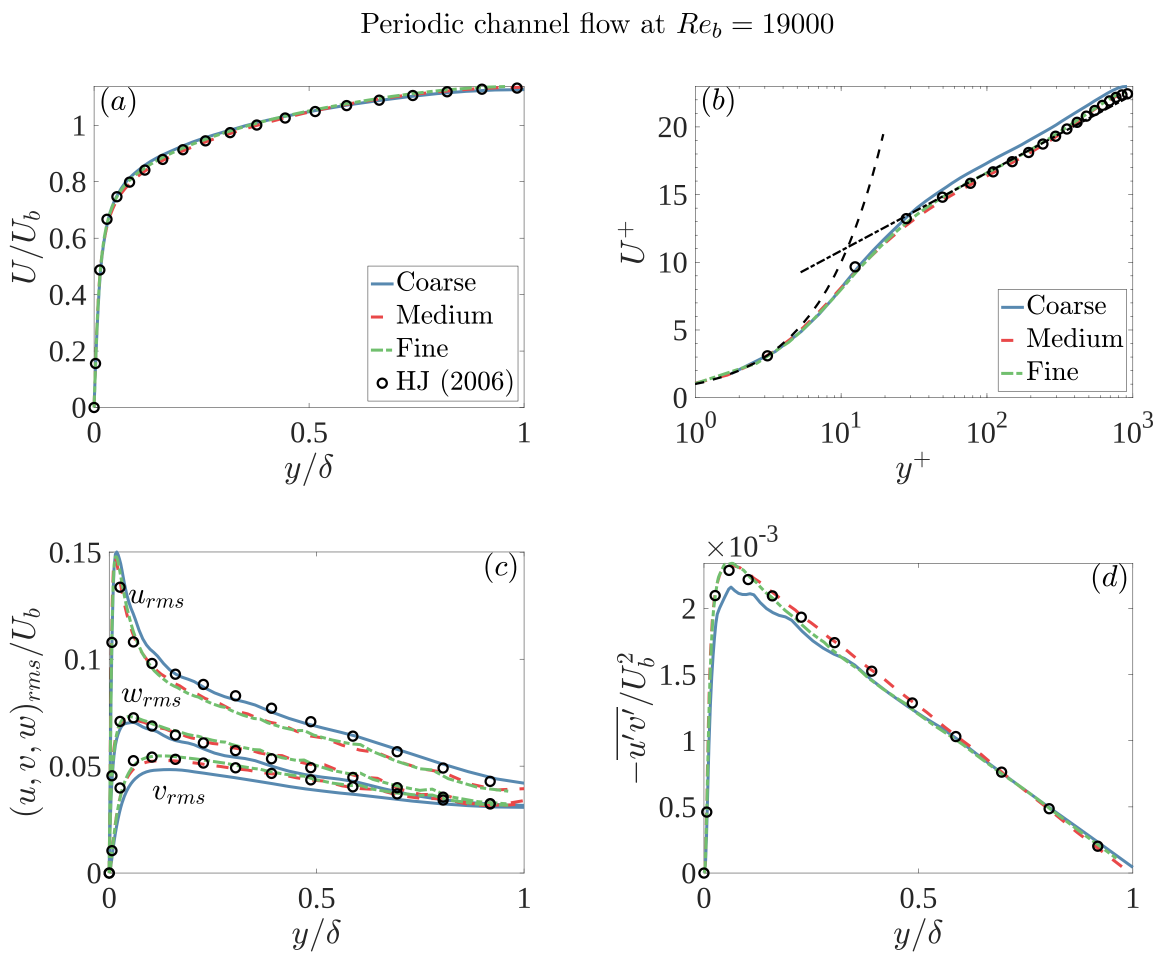

Figure 5: Grid convergence study for channel flow with SOD2D at $Re_\tau = 950$; *(a)* streamwise velocity in outer units; *(b)* streamwise velocity in inner units; *(c)* RMS of normal-component of turbulent stresses; *(d)* resolved Reynolds stresses

3.1.5 Results: GPU performance

We have explored a sequence of PDE-motivated bake-off problems designed to establish best practices for efficient high-order simulations across a variety of codes and platforms [53, 54]. As part of the benchmarking process, we are also investigating parallel performance for the two codes, NekRS and SOD2D. We tested strong-scaling performance on the Argonne Leadership Computing Facility (ALCF) Polaris, consisting of 560 nodes with 4 NVIDIA A100 GPUs per node.

For NekRS, we used a restart file at time 1000 as an initial condition with fully developed turbulent profile and ran for 1000 steps. In order to avoid initial transient behavior, we measured the average (wall) time per step, t_{step}, in seconds over steps 901-1000. We used a semi-implicit scheme (2nd-order backward differencing along the characteristics using 4th-order Runge-Kutta for the hyperbolic advection part with 1 substep for NekRS [55]), which permits $CFL = 1.7$ averaged over steps 901-1000. v_i and p_i are the average interation numbers per step over 901-1000 steps. For

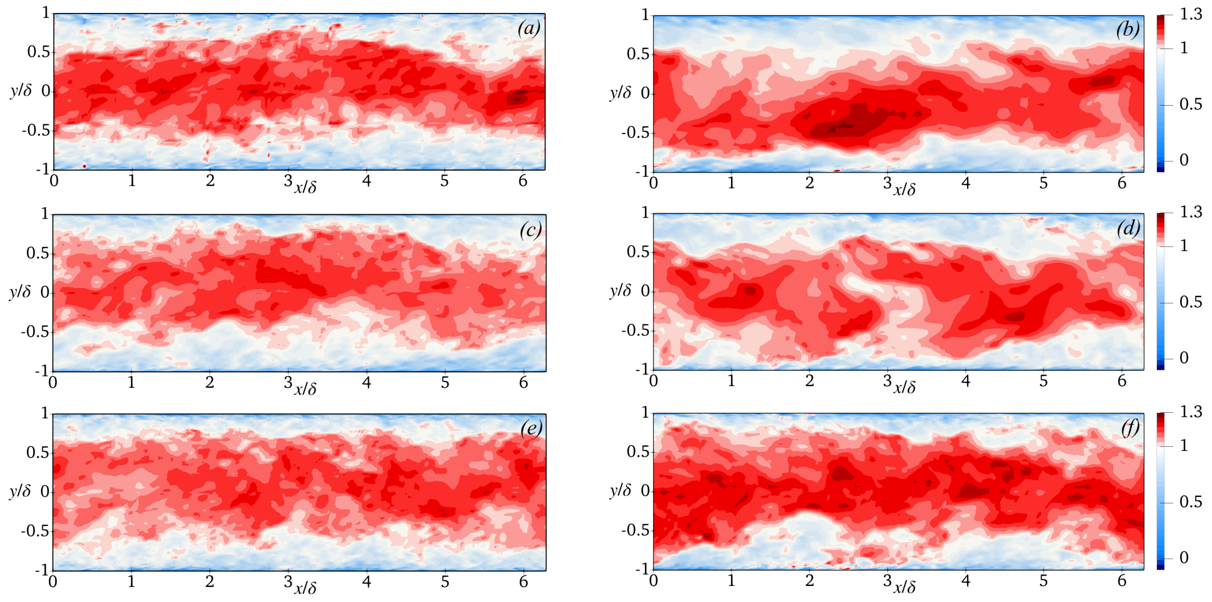

Figure 6: Visualization of instantaneous streamwise velocity (u/U_b) on a xy-plane for channel flow at $Re_\tau = 950$ for *(a,c,e)* Nek5000; *(b,d,f)* SOD2D; *(a,b)* Coarse resolution; *(c,d)* Medium resolution; *(e,f)* Fine resolution.

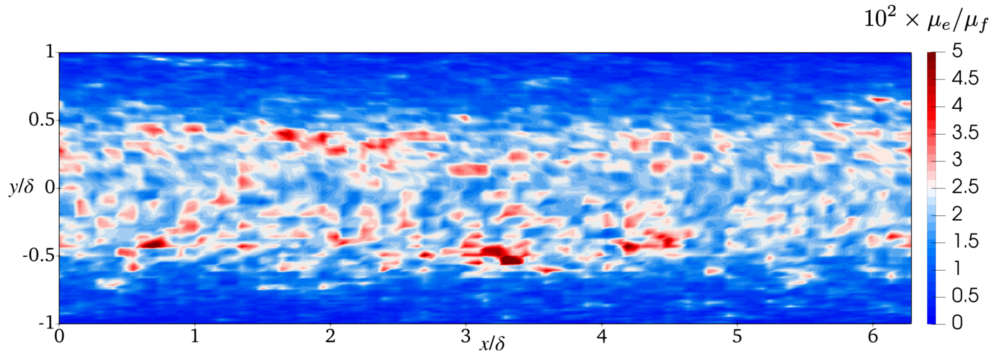

Figure 7: Visualization of instantaneous entropy-viscosity (μ_e/μ_f) on a xy-plane for channel flow at $Re_\tau = 950$ on Fine mesh for SOD2D.

SOD2D, we used a fully explicit Runge-Kutta 4th order time integration with $CFL = 1.5$. Starting from a fully-developed turbulent profile the simulations were ran for 10000 steps, measuring t_{step} every 10th step. Average of last 100 values thus collected were used to calculate the performance of the code.

Initial timing data is presented in Table 2 for a test case of channel flow with resolution 384^3 using 8–36 GPUs. We see that, for the case of 32 GPUs, there is an anomaly in the pressure iteration counts for NekRS that needs to be investigated, but the parallel behavior of the two codes is remarkably similar in terms of time per step.

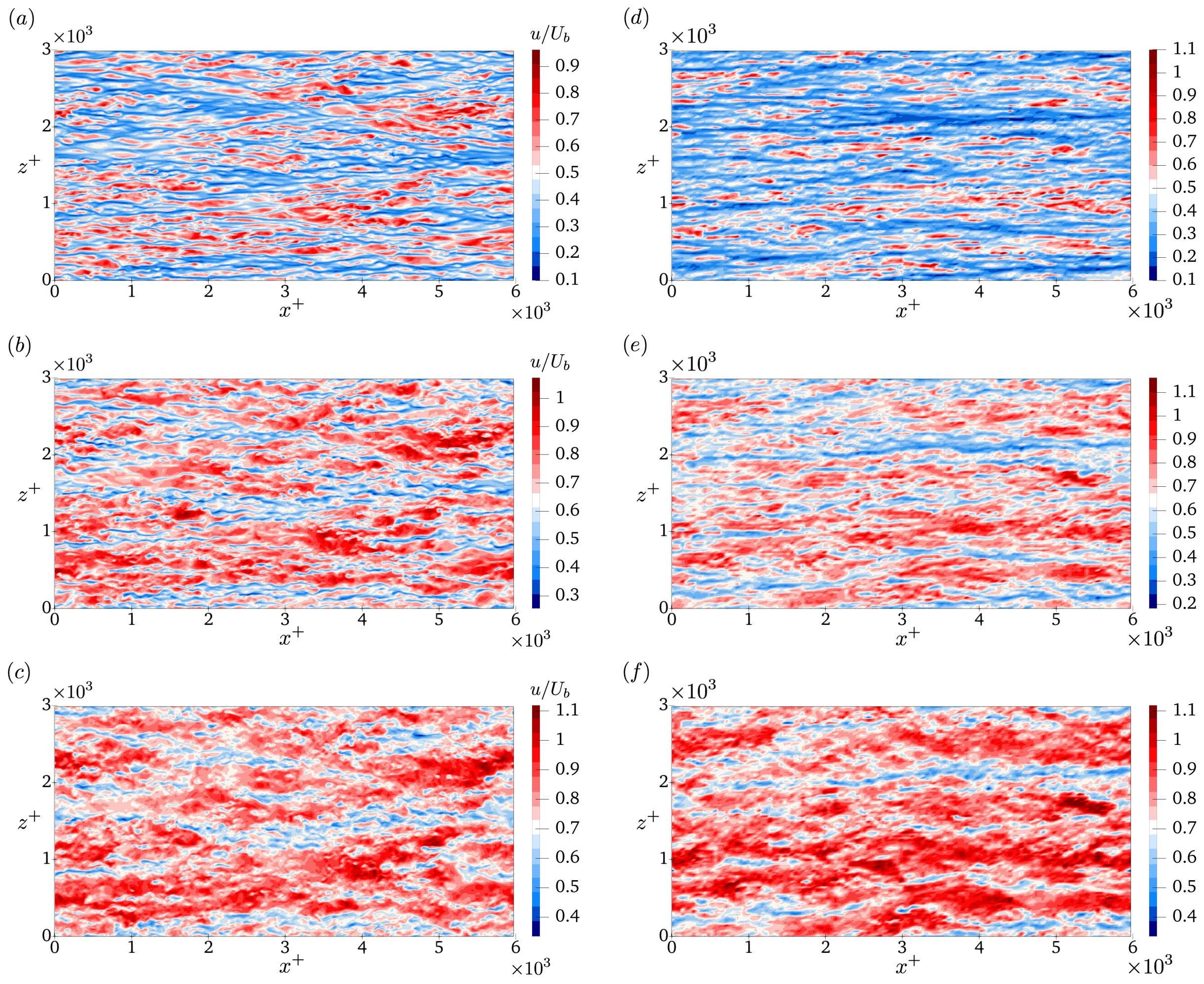

Figure 8: Visualization of instantaneous streamwise velocity (u/U_b), for channel flow at $Re_\tau = 950$, on wall-parallel planes at three different y^+ locations; *(top)* $y^+ = 11$; *(middle)* $y^+ = 30$; *(bottom)* $y^+ = 60$; *(a,c,e)* NekRS; *(b,d,f)* SOD2D.

3.2 Benchmark Case II: Periodic pipe flow

Next, we discuss our results for the V & V of turbulent pipe flow using Nek5000 and SOD2D using wall-resolved large-eddy simulations. We will be using the data of [56], who performed DNS at the same flow configuration, to conduct extensive validation for the results obtained from both the codes.

3.2.1 Simulation setup

We carried out simulations of turbulent flow in a circular pipe of radius R and an axial flow domain length $L_z = 12R$. Similar to the channel flow case in the previous section, we used periodic boundary condition in the axial (z) direction and impenetrable wall boundary conditions are

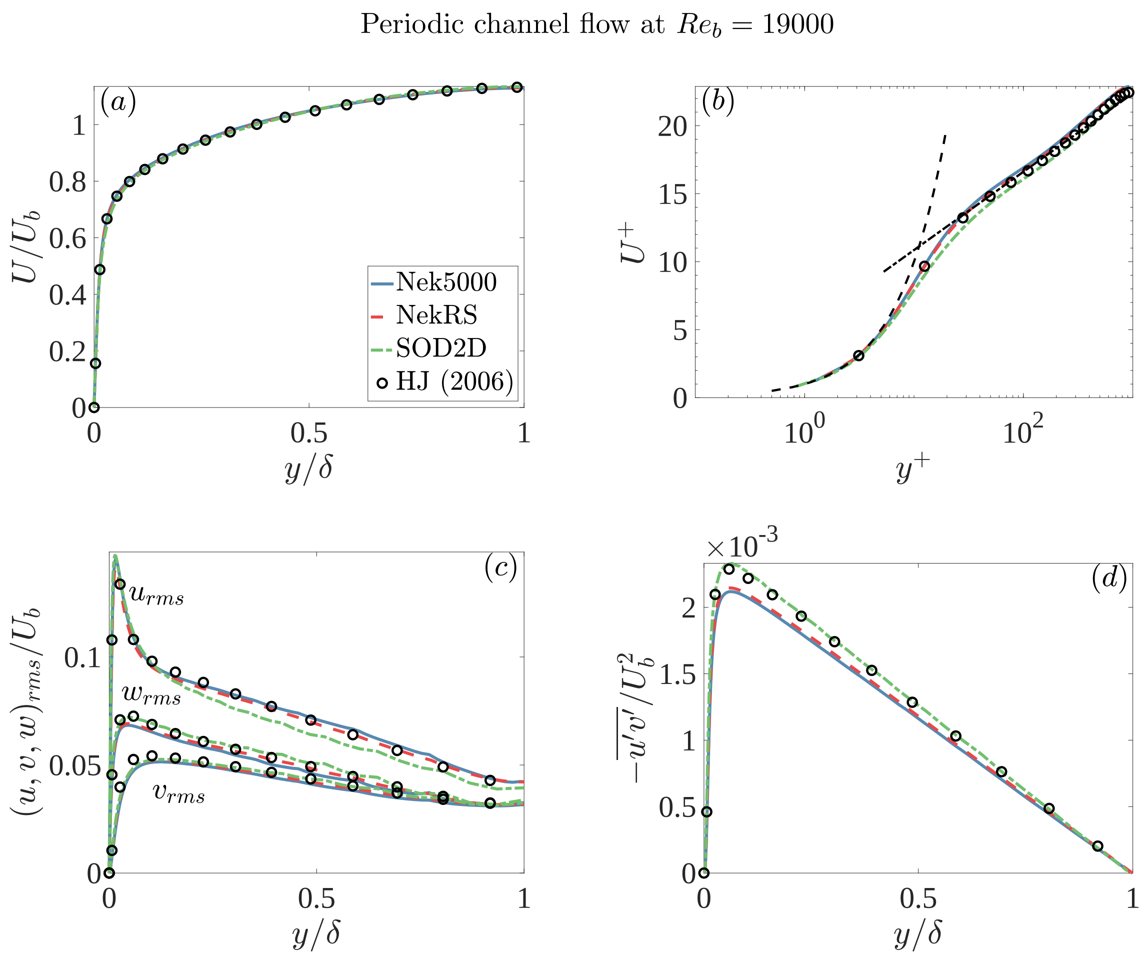

Figure 9: Statistics for channel flow at $Re_\tau = 950$; *(a)* streamwise velocity in outer units; *(b)* streamwise velocity in inner units; *(c)* RMS of normal turbulent stresses; *(d)* resolved Reynolds stresses.

imposed on the pipe boundaries. These simulations were conducted at a bulk Reynolds number $Re_b = 2RU_b/\nu = 37700$, which should yield a friction Reynolds number, $Re_\tau = u_\tau R/\nu = 1000$. Flow statistics were collected for a minimum of 1000 CTU for Nek5000 & 100 CTU for SOD2D, where $(1\,CTU = 2R/U_b)$, after reaching the steady-state. Together with temporal averaging, we have also averaged the flow in spanwise (z) and azimuthal (θ) directions. Since the mesh utilized (see Figure 10) does not conform in the θ-direction, the time-and-span averaged results were first interpolated on a $r-\theta$ structured mesh. The interpolation mesh utilized contains 97 & 385 points in r, θ directions respectively, and the grid was stretched in r-direction using a hyperbolic relation (similar to eqn. 44). The interpolation scheme utilized for NekRS and SOD2D, however, are different. NekRS utilizes a scheme (see [57] for details) that preserves the spectral accuracy of the code during grid-to-grid interpolation. For SOD2D, on the other hand, we are using only a second order accurate interpolation. As a consequence of using lower-order interpolation scheme, coupled with smaller statistics collection time, some loss in accuracy for SOD2D results when compared to NekRS is

Table 2: Performance of NekRS and SOD2D on Polaris. Tests have been performed for Channel flow at a resolution of $384 \times 384 \times 384$ with $E = 64^3$, $N = 6$, and the total number of grid points of 56 millions. $\Delta t = 1.5\text{e-}03$ ($CFL = 1.7$) and $\Delta t = 6.5\text{e-}05$ ($CFL = 1.5$) are used for NekRS and SOD2D, respectively.

	Performance on Polaris						
Code	nodes	GPUs	dofs/GPU	v_i	p_i	t_{step} (s)	P_{eff}
Nek5000	2	8	7.0779e+06	2.97	2.13	2.3512e-01	100
	3	12	4.7186e+06	2.97	2.14	1.7690e-01	88.6
	4	16	3.5389e+06	2.97	2.18	1.4138e-01	83.1
	5	20	2.8312e+06	2.97	2.09	1.2734e-01	73.8
	6	24	2.3593e+06	2.97	2.16	1.1305e-01	69.3
	8	32	1.7695e+06	2.97	5.16	1.3139e-01	44.7
	9	36	1.5729e+06	2.97	2.13	9.4724e-02	55.1
SOD2D	2	8	7.0779e+06	-	-	3.4603e-01	100
	3	12	4.7186e+06	-	-	2.4769e-01	93.1
	4	16	3.5389e+06	-	-	1.9941e-01	86.7
	5	20	2.8312e+06	-	-	1.6249e-01	85.1
	6	24	2.3593e+06	-	-	1.3526e-01	85.2
	8	32	1.7695e+06	-	-	1.0868e-01	79.5
	9	36	1.5729e+06	-	-	9.6119e-02	80.0

Table 3: Grid resolutions used for convergence study for the periodic pipe flow at $Re_\tau = 1000$; N_{xy} refers to the number of spectral elements at each cross-plane; N_p refers to the polynomial order used within the spectral elements; N_{dof} is the total number of grid points (in million, M); $\Delta r^+, \Delta(R\theta)^+, \Delta z^+$ denote spacings in viscous units; ϵ_{u_τ} is the error in u_τ/U_b, given by (47); Δt is the average simulation time-step.

	Vreman SGS model; $Re_\tau = 1000$							
Case	$N_{xy} \times N_z$	N_p	$N_{dof}\,(M)$	Δr^+	$\Delta(R\theta)^+$	Δz^+	ϵ_{u_τ}	$U_b\Delta t/D$
NekRS	96×14	7	0.4	4.8	56	122	-9.5	8.5×10^{-3}
	352×28		3.2	2.1	28	62	-1.5	3.8×10^{-3}
	1260×56		23.8	1.1	13	31	0.1	2.1×10^{-3}
SOD2D	96×14	6	0.35	5.6	65	142	9.3	2.7×10^{-4}
	352×28		2.1	2.5	33	71	-3.8	1.3×10^{-4}
	1260×56		15.2	1.2	14	35	-1.4	7.3×10^{-5}

expected. We will be working on utilizing high-order shape functions for interpolation in SOD2D in the future.

3.2.2 Grid convergence study

Before conducting the production runs a question that we addressed with these pipe flow simulations regards the optimal value of Smagorinsky parameter (C_s) to be used with the Vreman model. We

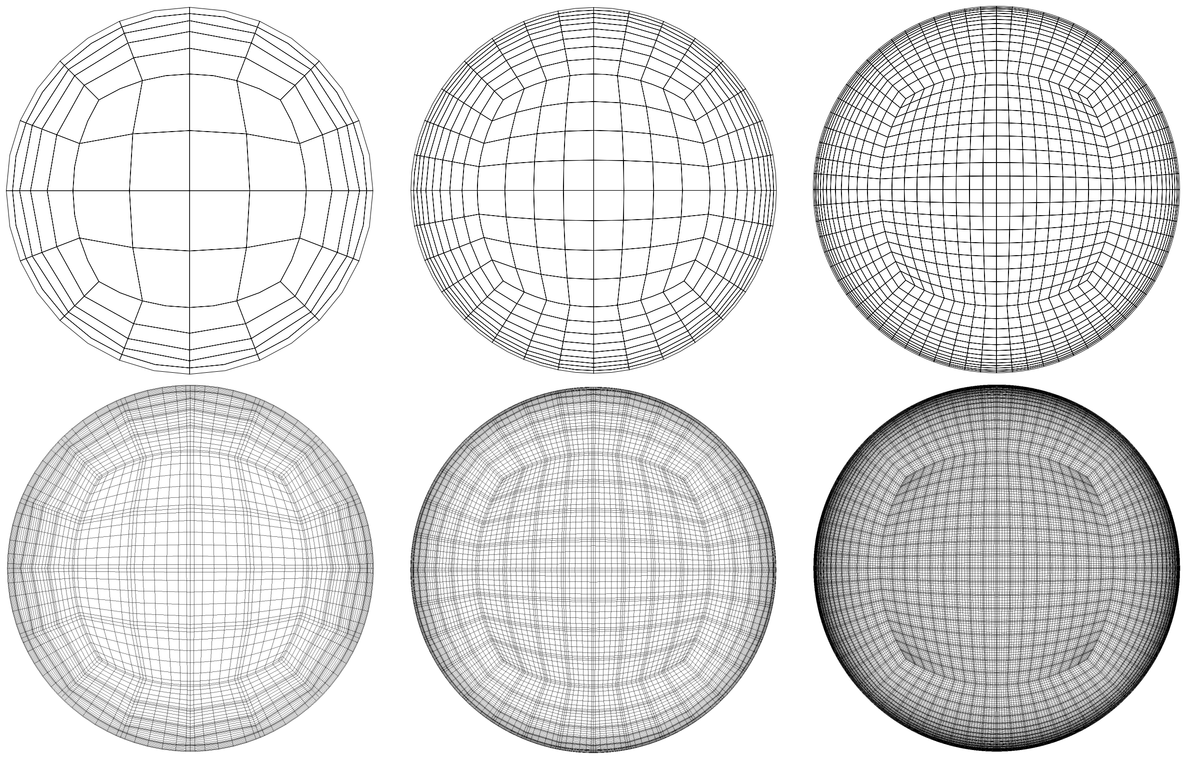

Figure 10: Cross section of spectral element meshes used for turbulent pipe flow simulations; *(a,d)* Coarse mesh; *(b,e)* Medium mesh; and *(c,f)* Fine mesh. The top row shows the element boundaries while the bottom row shows the Gauss-Lobatto-Legendre (GLL) collocation points for a polynomial order of $N = 6$. For resolution details refer to Table 3.

Table 4: Effect of Smagorinsky constant in Vreman model on flow statistics. These tests have been performed on the Medium grid resolution with NekRS for turbulent pipe flow mentioned in Table 3.

Case	SGS model	ϵ_{u_τ}	Re_τ
-	HPF	-0.8	991
$C_s = 0.2$	Vreman	-10.1	891
$C_s = 0.1$	Vreman	-3.9	959
$C_s = 0.065$	Vreman	-1.4	985
$C_s = 0.0325$	Vreman	0.8	1007
$C_s = 0.0$	Vreman	2.2	1020

tested various different values (listed in Table 4), together with the default setting of Nek5000/RS which uses a High-Pass Filter (HPF) to account for turbulent dissipation contribution from the unresolved scales. Figure 11 shows the effect of C_s on flow statistics for a Medium grid resolution. We clearly notice the strong impact of Smogorinsky parameter and, hence, point to the need to systematically fix it to an optimal value before the production runs. From Figure 11 we note that a value of $C_s \approx 0.065$ yields results that are consistent with the data of Khoury *et al.* [56], and

Periodic pipe flow at $Re_b = 38000$

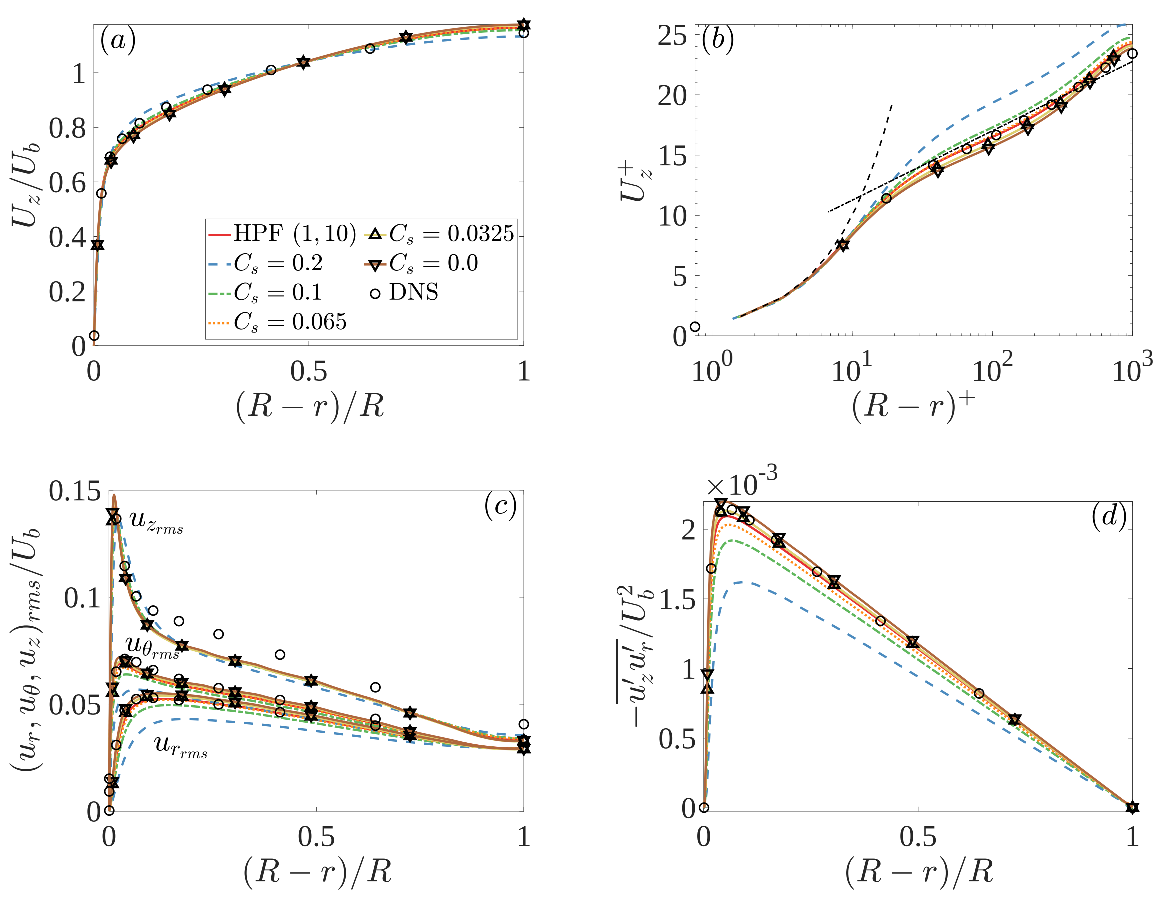

Figure 11: Effect of Smagorinsky model parameter (C_s) on turbulent pipe flow using NekRS at $Re_\tau = 1000$; *(a)* streamwise velocity (U_z/U_b) in outer units; *(b)* streamwise velocity in inner units; *(c)* normal stress components; and *(d)* resolved Reynolds stresses

hence we used this value throughout all the pipe flow simulations with NekRS. The effect of C_s on WRLES with SOD2D, however, was not conducted and would be a topic of our future work. It would be interesting to see how changes in C_s affects the compressible flow variables within an entropy-viscosity formulation. For this we use $C_s = 0.1$ with SOD2D.

Figures 12–13 show the results obtained for the grid refinement study for Nek5000 and SOD2D respectively. We notice that the mean axial velocity (U_z/U_b) collapses nicely with the data of [56], both in the outer (Figure 12(a)) and inner (Figure 12(b)) units, for Medium and Fine grids for the two codes. The error in u_τ is also reported in Table 3. Since we are using much coarser resolution than channel flow in the previous section, u_τ erros are more pronounced for both Medium and Fine grid but still under 5% when compared to DNS.

Figures 12–13*(c,d)* show the resolved root-mean-square (rms) value of normal Reynolds fluctuations in polar coordinates $(u_r, u_\theta, u_z)_{rms}$, and resolved Reynolds-shear stress (RSS) $(\overline{u_z' u_r'})$. They

Periodic pipe flow at $Re_b = 37700$

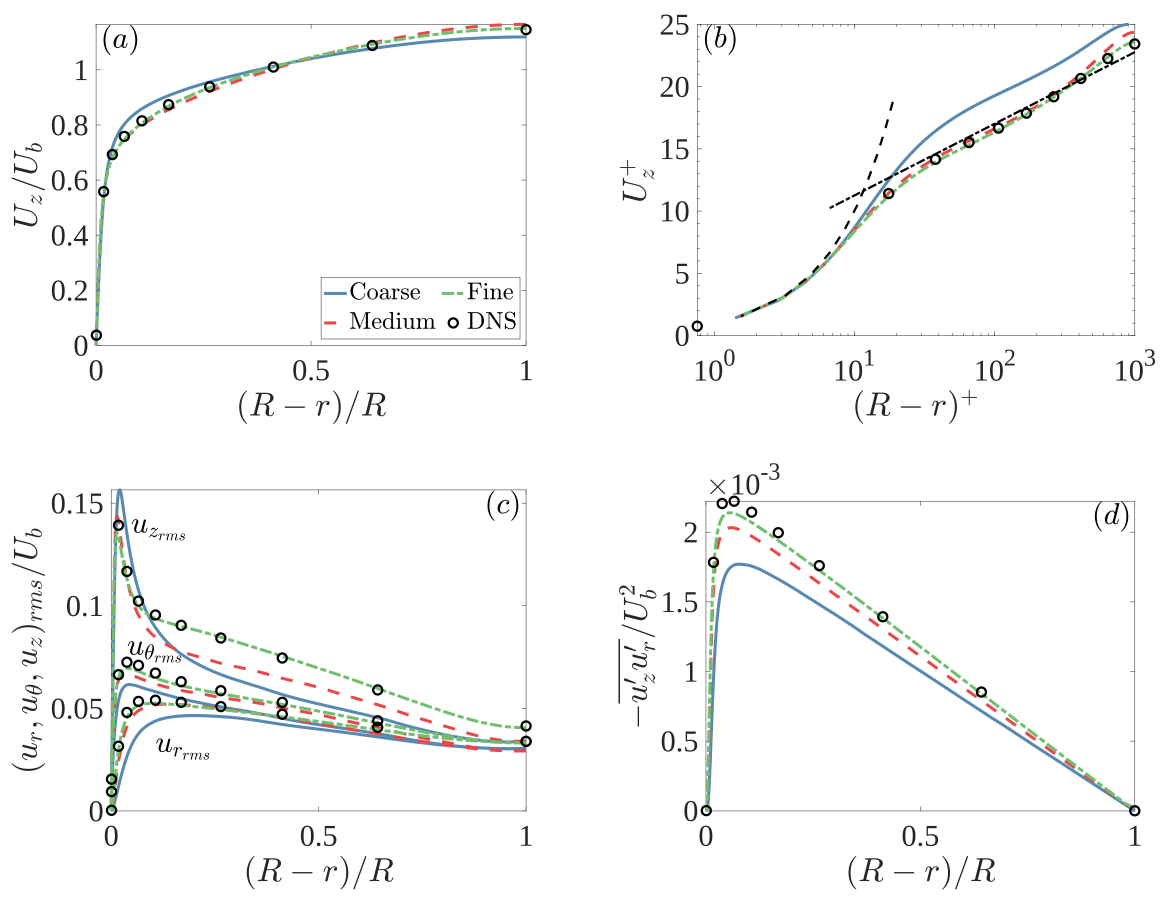

Figure 12: Grid convergence study for pipe flow using NekRS at $Re_\tau = 1000$; *(a)* streamwise velocity (U_z/U_b) in outer units; *(b)* streamwise velocity in inner units; *(c)* normal turbulent stresses; *(d)* resolved Reynolds stress.

are defined as

$$u_{r_{rms}} = \sqrt{\overline{u_r' u_r'}}; \quad u_{\theta_{rms}} = \sqrt{\overline{u_\theta' u_\theta'}}; \quad u_{z_{rms}} = \sqrt{\overline{u_z' u_z'}} \tag{49}$$

Overall, the corroboration between our results and the DNS data is adequate for Medium and Fine grids and, hence, all the results discussed henceforth are for the Medium grid resolution only.

3.2.3 Flow visualization

Figure 14 shows the instantaneous axial velocity for the two CFD codes utilized in this study, at the three resolution levels reported in Table 3. Similar to the channel flow case, we observe the mesh imprints at the Coarse resolution level in NekRS (Figure 14*(a)*), which is not visible at the Medium and Fine resolutions(Figure 14*(c,e)*). The finer turbulent fluctuations of SOD2D (Figure 14*(b,d,f)*)

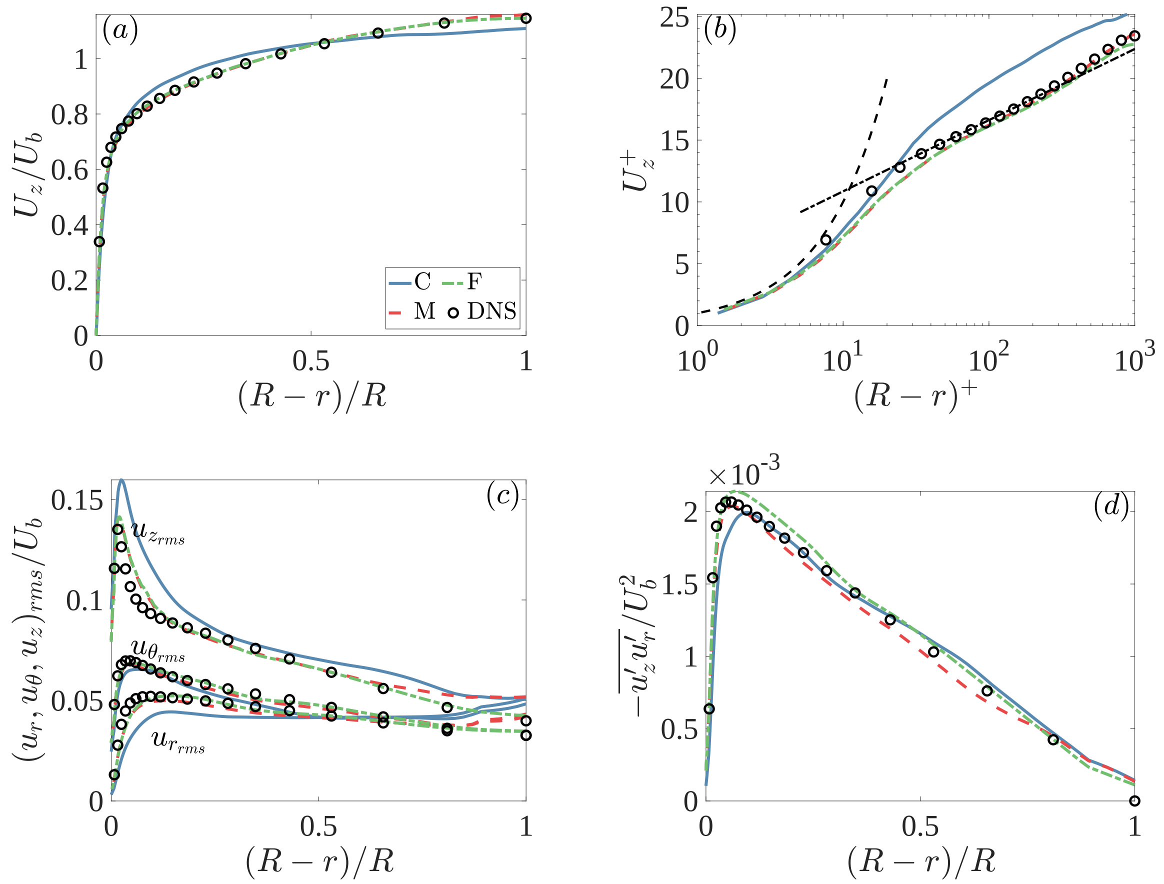

Figure 13: Grid convergence study for pipe flow using SOD2D at $Re_\tau = 1000$; *(a)* axial velocity (U_z/U_b) in outer units; *(b)* axial velocity in inner units; *(c)* normal turbulent stresses; *(d)* resolved Reynolds stress.

look similar to that of NekRS at all resolution levels.

3.2.4 Flow statistics

Figure 15 compares the statistical results of NekRS and SOD2D for pipe flow Medium grid resolution mentioned in Table 1. Since we have utilized a coarser resolution at each level for the pipe flow (when compared to channel flow in Section 3.1.1), we observe some difference in the Medium and the Fine resolution, particularly for the 2$^{\text{nd}}$ order statistics which have strict requirements for resolution. Nonetheless, our aim to perform V & V study at reduced resolution levels are considered met with these results as the results look more accurate at successive refinement levels for NekRS.

Figure 15 compares the results for SOD2D with Nek5000/RS at the Medium grid resolution for various statistical quantities. We observed very good match in the first-order statistics. There are some discrepancies in the 2$^{\text{nd}}$ order statistics, which seem related to low statistics collection

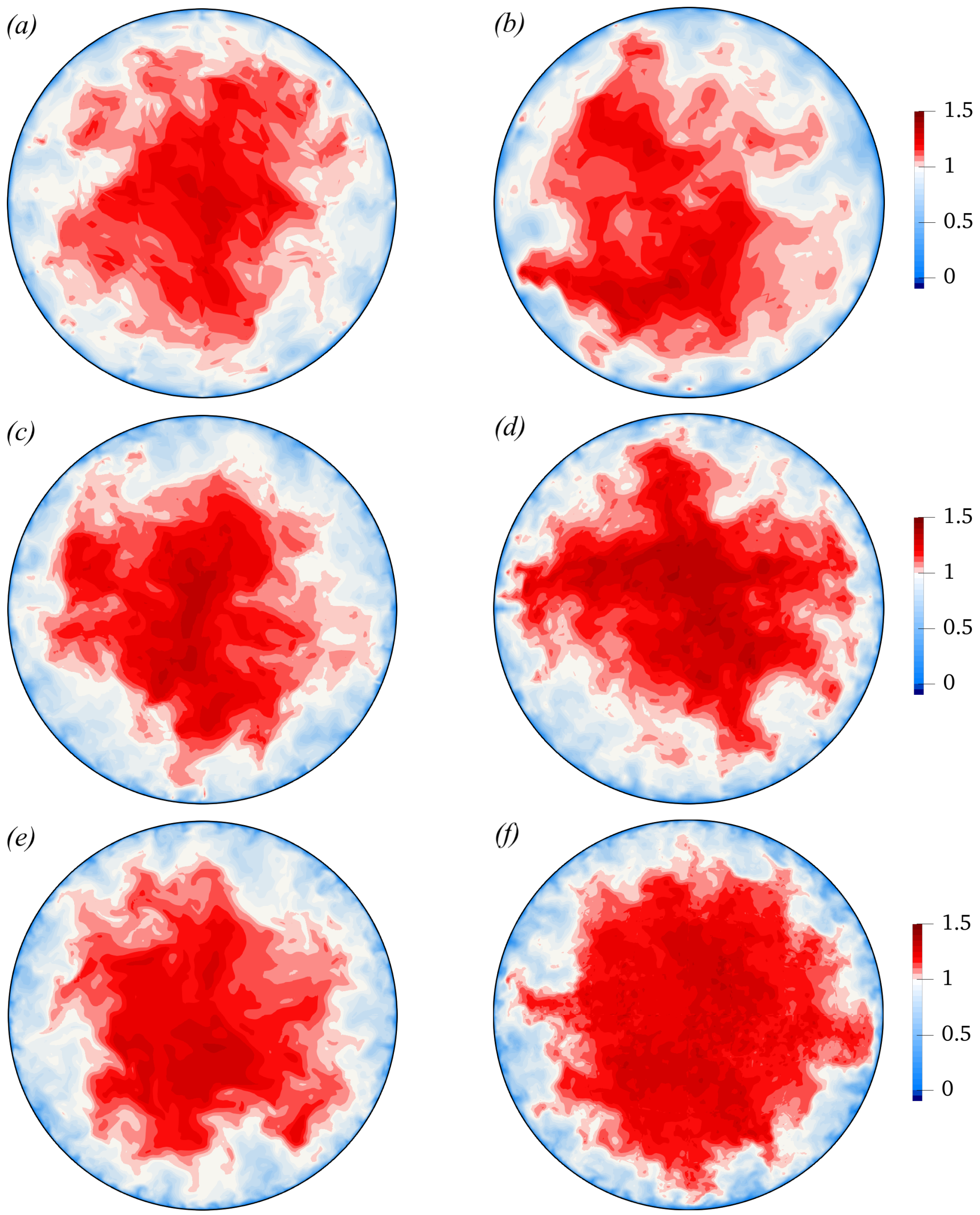

Figure 14: Visualization of instantaneous axial velocity (u_z/U_b) for pipe flow at $Re_\tau = 1000$; *(a,c,e)* Nek5000; *(b,d,f)* SOD2D; *(a,b)* Coarse resolution; *(c,d)* Medium resolution; *(e,f)* Fine resolution; mesh resolution are indicated in Table 3.

time for SOD2D as mentioned in the previous section.

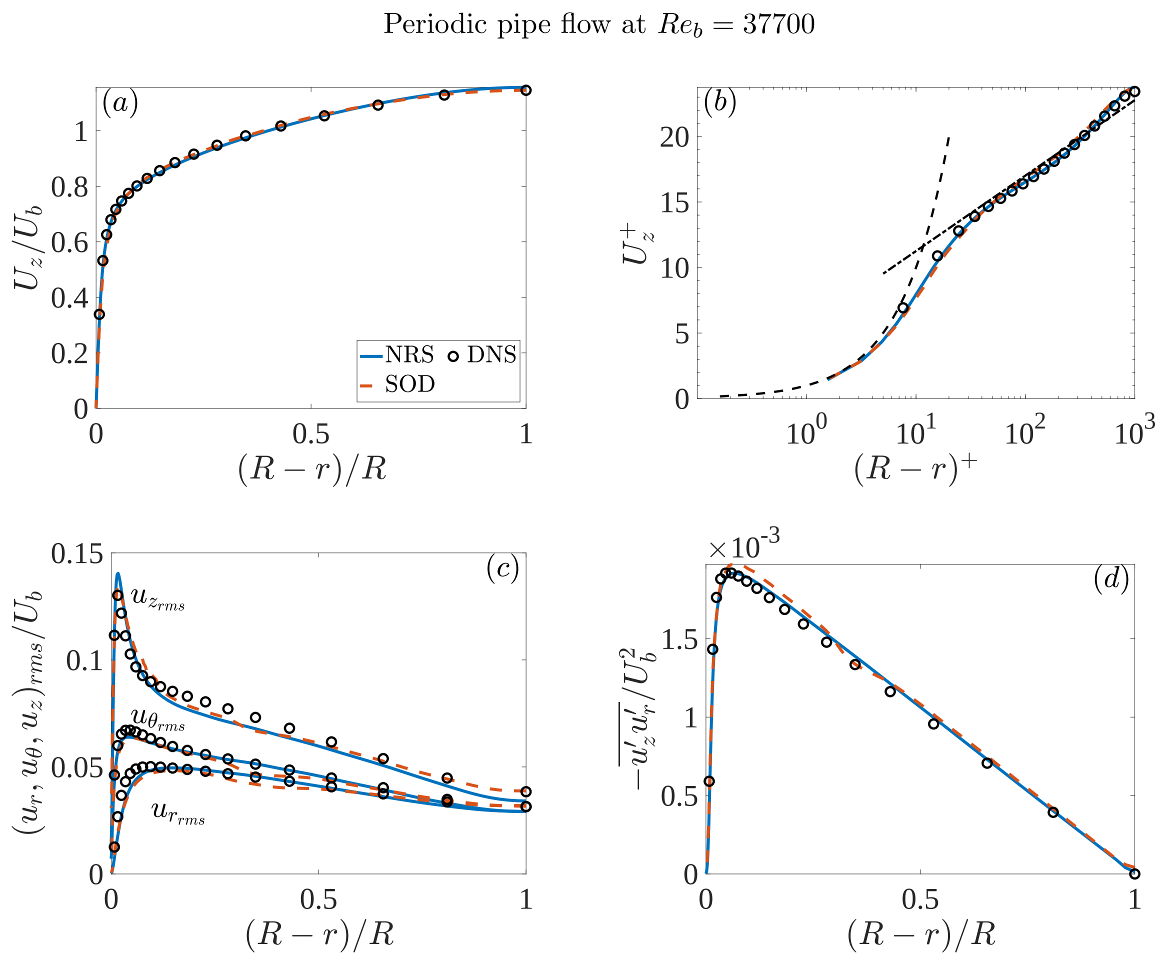

Figure 15: Comparison of turbulent pipe flow results between NekRS and SOD2D at $Re_\tau = 1000$ on the Fine grid resolution; *(a)* axial velocity (U_z/U_b) in outer units; *(b)* axial velocity in inner units; *(c)* normal turbulent stresses; *(d)* resolved Reynolds stress.

3.3 Implementation and testing of wall-models for LES in Nek5000

In this final section, we report on our initial effort to implement wall-models for large-eddy simulations. We will be using benchmark problem of periodic channel flow for the validation part. The aim was to implement both the conventional as well as machine-learning wall-models in Nek5000. For the conventional wall-modeling, we implemented the standard and Reichardt's variant of the log-law models described in detail in Section 2. For the machine-learning part, we followed the approach of [41], to implement decision-tree based wall-model. One specific aim of this study was to answer the following question: *can machine-learning wall-models trained on the data-set of one CFD code be used with another CFD code, which might be potentially dissimilar in their numerical formulation?*. Towards that, we will be using the trained decision-tree based ML model from [41], who used a low-order finite element code Alya (described in detail in Section Section 2), to perform wall-model validation tests with spectral code Nek5000 on periodic channel flow.

Results of wall-modeling reported in this section have been achieved by simulating channel flow in a box of dimension $2\pi\delta \times 2\delta \times \pi\delta$, similar to one reported for WRLES in Section 3.1.1. For Nek5000 we use $16 \times 16 \times 16$ elements to discretize the domain, with a polynomial order, $p = 6$. Here, there are two main differences from the wall-resolved LES cases. First, the grid in the wall-normal direction remains uniform. And second, instead of a no-slip boundary condition, we employ a slip boundary condition on the walls. For more details refer to Section 2. Since Alya uses basic finite-elements, we use $96 \times 96 \times 96$ quadrilateral elements to discretize the same domain size mentioned above.

Figure 16 shows the effect of sampling location on the performance of the wall-model for Nek5000. Here, we are using the standard log-law (with $\kappa = 0.4$, $B = 5.0$). We note that the differences in wall-models results for different sampling locations are apparent in inner scales . For sampling location too close to the wall (red line in Figure 16*(b)*), we see the log-law mismatch clearly. As we sample farther away from the wall, the log-law mismatch decreases significantly (two tested locations at $y^+ = 60$ and $y^+ = 190$ in Figure 16). We finally note that the wall-model results presented here show enhanced accuracy than the corresponding wall-resolved calculation with the same grid density (note the grids in wall-resolved calculations are stretched). This could be due to the increased resolution of the wall-model case in the centre of the channel as it uses grid with uniform spacing.

Before presenting the results of machine learning WM for Nek5000, we also carried out a verification study for Alya with different wall-model and a wall-resolved calculation. The sampling location for wall-model calculations with Alya is fixed at 3rd grid location (corresponding to $y^+ \approx 56$). The results are shown in Figure 17*(a)*. We see that both the log-law (Reichardt's law) and the ML wall-models yield very good results, and show minimal log-law mismatch. This behavior is consistent with the recommendation on wall-modeling in literature [39, 3].

Figure 17*(b)* shows the initial tests performed with ML wall-models in Nek5000, using the decision tree model trained on the data of Alya. We observe that for low learning parameter (red-dashed line in the figure) Nek5000 does not predict the correct log-law behaviour of the flow, whereby the slope of the prediction does not match the correct value. This is in contrast to Alya which predicts the behavior quite accurately. The predictions of Nek5000 improve as the learning rate is increased. However, the accuracy of the ML wall-model in Nek5000 saturates with increasing learning rate and there is still significant log-law mismatch in the results. This is also visible at the instantaneous level in Figure 18, where we observed significant damping of velocity fluctuations in Nek5000 when compared to Alya. The streaks are also less pronounced in Nek5000(Figure 18*(c)*) than in Alya (Figure 18*(d)*). One explanation for this difference in behaviour between Nek5000 and Alya could be the difference in approach of applying the boundary condition for wall-models for the two codes. As mentioned previously, the boundary condition in Alya is applied at each boundary node using the local interpolated value of the velocity. For Nek5000, on the other hand, we apply the boundary condition using a single interpolated value of interpolated velocity. Decision tree by there very design work on classification and regression of input data. Hence, it might be worthwhile to check their performance in Nek5000 if local velocity array, instead of a single value, is input to the ML model. This will be addressed in our future work.

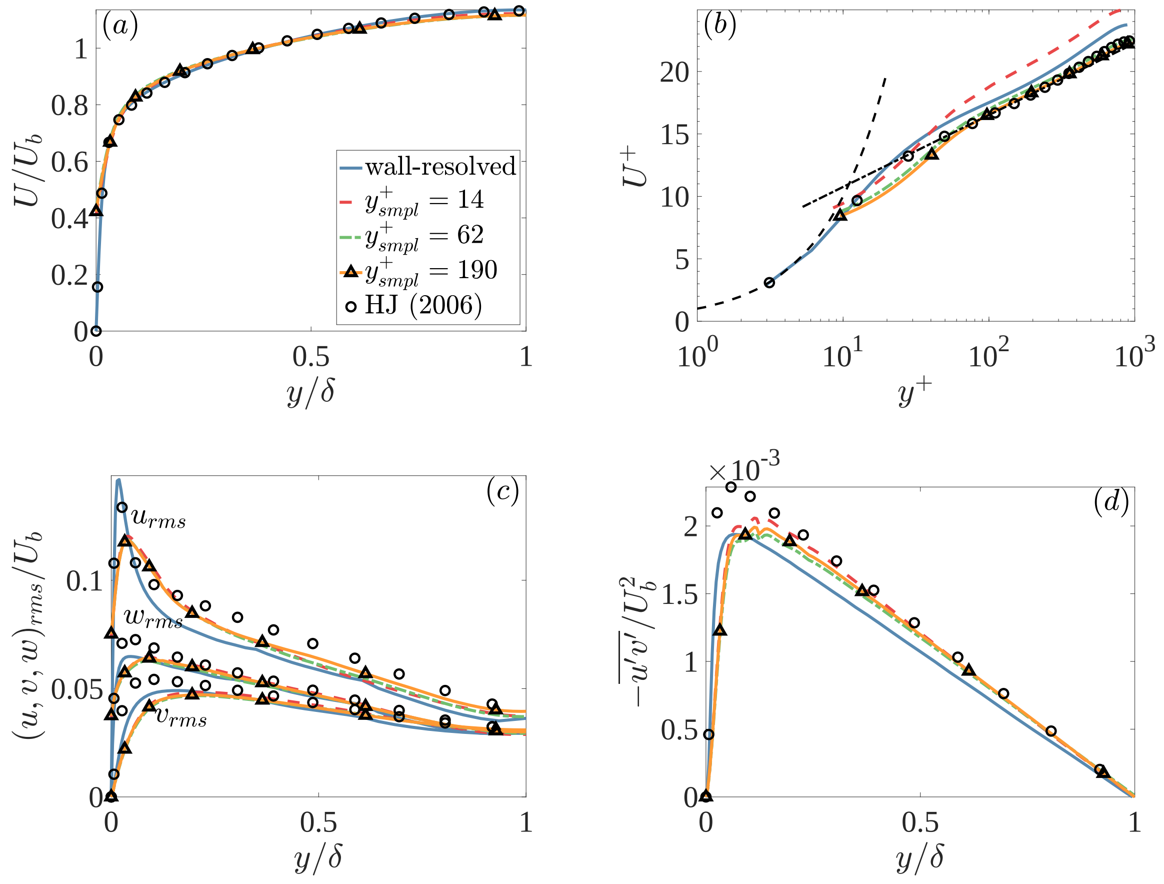

Figure 16: Effect of sampling location on WMLES results for channel flow at $Re_\tau = 950$ with Nek5000; *(a)* streamwise velocity in outer units; *(b)* streamwise velocity in inner units; *(c)* RMS of normal turbulent stresses; *(d)* resolved Reynolds stresses

4 Conclusion

This study was aimed at two specific goals: first, to validate and verify two spectral-element based CFD solvers, Nek5000/RS and SOD2D. Nek5000, and its GPU variant NekRS, is aimed at solving incompressible flow problems with high accuracy and has been a standard in turbulence research for many years. SOD2D, on the other hand, focuses on solving compressible fluid flow and uses an entropy-visocsity formulation to capture flow discontinuities. We performed extensive validation of the two codes for two benchmark turbulent flow configurations: periodic channel and pipe flows. We demonstrated the accuracy of these codes through full statistical analysis, grid convergence and turbulent structure analysis using wall-resolved large eddy simulations (WRLES).

In the second part of the study, we reported on our initial effort to implement and validate wall-modeling strategies for large-eddy simulations in Nek5000. These included both the conventional, log-law based wall models as well as decision-tree based machine-learning wall model. The question

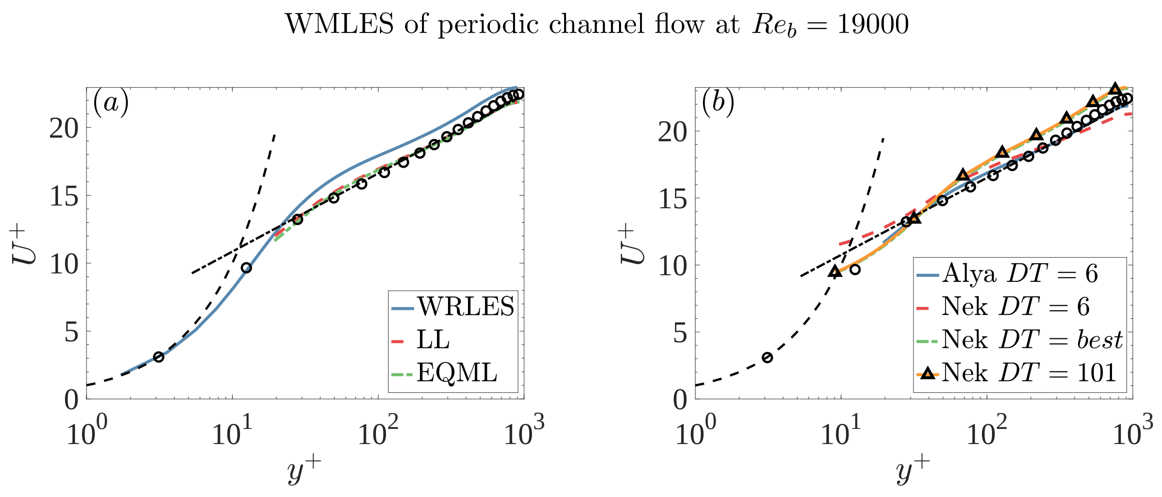

Figure 17: Comparison of wall-modeled large-eddy results for channel flow at $Re_\tau = 950$ for *(a)* Alya; *(b)* Nek5000 showing streamwise velocity in inner units; sampling location, $y_s^+ \approx 60$ for both the codes; DT refers to number of decision tress that the ML model employs for learning.

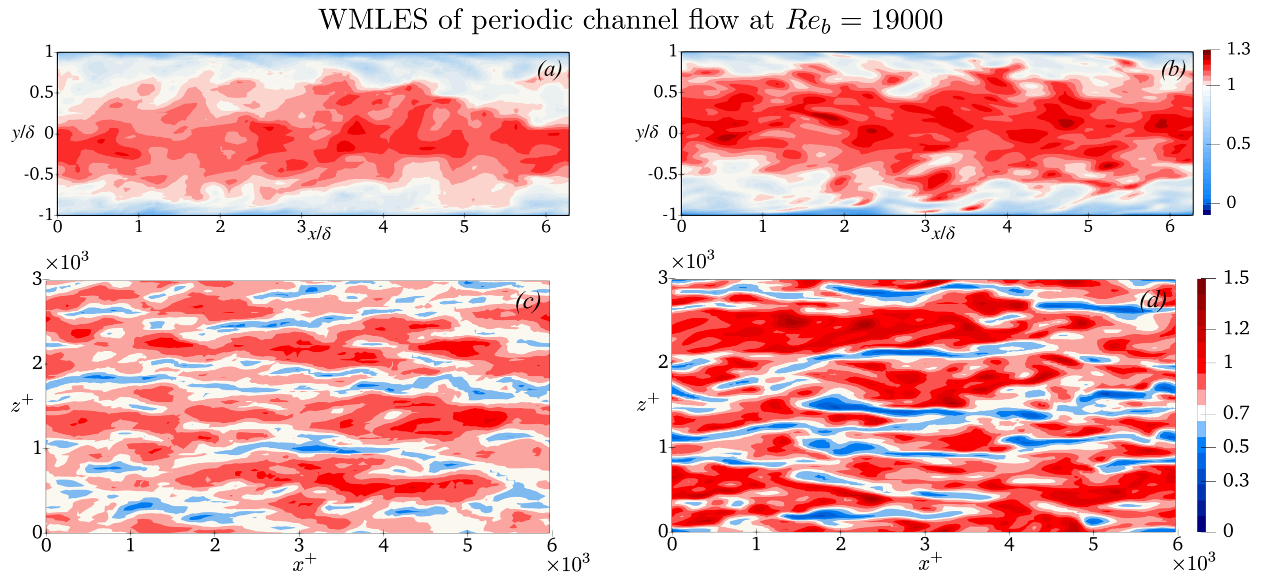

Figure 18: visualization of instantaneous streamwise velocity (u/U_b) on *(a,b)* $x - y$ plane; *(c,d)* wall-parallel plane at $y^+ = 60$; *(a,c)* Nek5000 and *(b,d)* Alya for ML based wall-model.

of interest for ML model is to check the accuracy of a ML wall-model trained on the data-set of a low-order finite element based code in Nek5000/RS. We found that the results for the log-law based wall-model agree very well with the resolved simulations, particularly for sampling locations deep in the log layer. The results for ML wall-model, however, show significant log-law mismatch and need further testing. As pointed out, the sampling strategy in Nek5000 might be problematic within the context of decision-tree based wall modeling, and would be explored in detail in our future studies.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357. This work is authored in part by a Ramń y Cajal postdoctoral contract by the Ministerio de Economá y Competitividad, Secretará de Estado de Investigación, Desarrollo e Innovación, Spain (RYC2018-025949-I). This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

[1] Haecheon Choi and Parviz Moin. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Physics of Fluids*, 24(1):011702, 1 2012.

[2] Xiang I.A. Yang and Kevin P. Griffin. Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation. *Physics of Fluids*, 33(1), 2021.

[3] Ugo Piomelli. Wall-layer models for large-eddy simulations. *Progress in Aerospace Sciences*, 44(6):437–446, 2008.

[4] Sanjeeb T. Bose and George Ilhwan Park. Wall-Modeled Large-Eddy Simulation for Complex Turbulent Flows. *Annual Review of Fluid Mechanics*, 50(1):535–561, 2018.

[5] Elias Balaras, Carlo Benocci, and Ugo Piomelli. Two-layer approximate boundary conditions for large-eddy simulations. *AIAA Journal*, 34(6):1111–1119, 6 1996.

[6] Johan LARSSON, Soshi KAWAI, Julien BODART, and Ivan BERMEJO-MORENO. Large eddy simulation with modeled wall-stress: recent progress and future directions. *Mechanical Engineering Reviews*, 3(1):15–00418, 2016.

[7] A. Frère, K. Hillewaert, P. Chatelain, and G. Winckelmans. High Reynolds Number Airfoil: From Wall-Resolved to Wall-Modeled LES. *Flow, Turbulence and Combustion*, 101(2):457–476, 9 2018.

[8] M.O. Deville, P.F. Fischer, and E.H. Mund. *High-order methods for incompressible fluid flow.* Cambridge University Press, Cambridge, 2002.

[9] Lucas Gasparino Ferreira da Silva, Filippo Spiga, and Oriol Lehmkuhl. SOD2D: An open-source GPU-enabled spectral finite elements method for compressible scale-resolving simulations. https://gitlab.com/bsc_sod2d/sod2d_gitlab, 2023.

[10] Paul Fischer, Stefan Kerkemeier, Misun Min, Yu-Hsiang Lan, Malachi Phillips, Thilina Rathnayake, Elia Merzari, Ananias Tomboulides, Ali Karakus, Noel Chalmers, and Tim Warburton. Nekrs, a gpu-accelerated spectral element navier-stokes solver. *CoRR*, abs/2104.05829, 2021.

[11] Misun Min, Yu-Hsiang Lan, P. Fischer, E. Merzari, S. Kerkemeier, , M. Phillips, T. Rathnayake, A. Novak, D. Gaston, N. Chalmers, and T. Warburton. Optimization of full-core reactor simulations on summit. *2022 International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2022. accepted.

[12] P. Fischer, J. Lottes, and S. Kerkemeier. Nek5000: Open source spectral element CFD solver. http://nek5000.mcs.anl.gov and https://github.com/nek5000/nek5000. 2008.

[13] P. Fischer, K. Heisey, and M. Min. Scaling limits for PDE-based simulation (invited). In *22nd AIAA Computational Fluid Dynamics Conference, AIAA Aviation.* AIAA 2015-3049, 2015.

[14] A.T. Patera. A spectral element method for fluid dynamics : laminar flow in a channel expansion. *J. Comput. Phys.*, 54:468–488, 1984.

[15] S.A. Orszag. Spectral methods for problems in complex geometry. *J. Comput. Phys.*, 37:70–92, 1980.

[16] P. Fischer, M. Schmitt, and A. Tomboulides. *Recent developments in spectral element simulations of moving-domain problems*, volume 79, pages 213–244. Fields Institute Communications, Springer, 2017.

[17] S. Stolz, P. Schlatter, and L. Kleiser. High-pass filtered eddy-viscosity models for large-eddy simulations of transitional and turbulent flow. *Physics of Fluids*, 17(6):065103, 2005.

[18] P.F. Fischer and J.W. Lottes. Hybrid Schwarz-multigrid methods for the spectral element method: Extensions to Navier-Stokes. In R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu, editors, *Domain Decomposition Methods in Science and Engineering Series.* Springer, Berlin, 2004.

[19] M. Phillips, S. Kerkemeier, and P. Fischer. Tuning spectral element preconditioners for parallel scalability on GPUs. In *Proc. of the 2022 SIAM Conf. on Par. Proc. for Sci. Comp.*, pages 37–48. SIAM, 2022.

[20] Jean-Luc Guermond, Richard Pasquetti, and Bojan Popov. Entropy viscosity method for nonlinear conservation laws. *Journal of Computational Physics*, 230(11):4248–4267, 5 2011.

[21] Mariano Vázquez, Guillaume Houzeaux, Seid Koric, Antoni Artigues, Jazmin Aguado-Sierra, Ruth Arís, Daniel Mira, Hadrien Calmet, Fernando Cucchietti, Herbert Owen, Ahmed Taha, Evan Dering Burness, José María Cela, and Mateo Valero. Alya: Multiphysics engineering simulation toward exascale. *Journal of Computational Science*, 14:15–27, 5 2016.

[22] G. Houzeaux, M. Vázquez, R. Aubry, and J. M. Cela. A massively parallel fractional step solver for incompressible flows. *Journal of Computational Physics*, 228(17):6316–6332, 2009.

[23] S. Gövert, D. Mira, J.B.W. Kok, M. Vázquez, and G. Houzeaux. Turbulent combustion modelling of a confined premixed jet flame including heat loss effects using tabulated chemistry. *Applied Energy*, 156:804–815, 10 2015.

[24] S. Gövert, D. Mira, M. Zavala-Ake, J.B.W. Kok, M. Vázquez, and G. Houzeaux. Heat loss prediction of a confined premixed jet flame using a conjugate heat transfer approach. *International Journal of Heat and Mass Transfer*, 107:882–894, 4 2017.

[25] I Rodriguez, R Borell, O Lehmkuhl, C. D. Perez Segarra, and A Olivia. Direct numerical simulation of the flow over a sphere at Re = 3700. *Journal of Fluid Mechanics*, 679:263–287, 7 2011.

[26] D. E. Aljure, O. Lehmkuhl, I. Rodríguez, and A. Oliva. Flow and turbulent structures around simplified car models. *Computers and Fluids*, 2014.

[27] Oriol Lehmkuhl, Ugo Piomelli, and Guillaume Houzeaux. On the extension of the integral length-scale approximation model to complex geometries. *International Journal of Heat and Fluid Flow*, 78(May):108422, 2019.

[28] Oriol Lehmkuhl, Adrián Lozano-Durán, and Ivette Rodriguez. Active flow control for external aerodynamics: from micro air vehicles to a full aircraft in stall. *Journal of Physics: Conference Series*, 1522(1):012017, 4 2020.

[29] Sergey Charnyi, Timo Heister, Maxim A. Olshanskii, and Leo G. Rebholz. On conservation laws of Navier–Stokes Galerkin discretizations. *Journal of Computational Physics*, 337:289–308, 5 2017.

[30] F. X. Trias and O. Lehmkuhl. A Self-Adaptive Strategy for the Time Integration of Navier-Stokes Equations. *Numerical Heat Transfer, Part B: Fundamentals*, 60(2):116–134, 8 2011.

[31] Rainald Löhner, Fernando Mut, Juan Raul Cebral, Romain Aubry, and Guillaume Houzeaux. Deflated preconditioned conjugate gradient solvers for the pressure-Poisson equation: Extensions and improvements. *International Journal for Numerical Methods in Engineering*, 87(1-5):2–14, 7 2011.

[32] A. W. Vreman. An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications. *Physics of Fluids*, 16(10):3670–3681, 10 2004.

[33] Ronald J. Adrian and Parviz Moin. Stochastic estimation of organized turbulent structure: homogeneous shear flow. *Journal of Fluid Mechanics*, 190:531–559, 5 1988.

[34] S. Stolz, P. Schlatter, and L. Kleiser. High-pass filtered eddy-viscosity models for large-eddy simulations of transitional and turbulent flow. *Physics of Fluids*, 17(6):1–14, 2005.

[35] Claudio Canuto. Stabilization of spectral methods by finite element bubble functions. *Computer Methods in Applied Mechanics and Engineering*, 116:13–26, 1 1994.

[36] P. Schlatter, S. Stolz, and L. Kleiser. Analysis of the sgs energy budget for deconvolution-and relaxation-based models in channel flow. In *Direct and Large-Eddy Simulation VI*, pages 135–142. Springer, Dordrecht, 2006.

[37] Philipp Schlatter, Steffen Stolz, and Leonhard Kleiser. Les of transitional flows using the approximate deconvolution model. *International Journal of Heat and Fluid Flow*, 25(3):549–558, 2004. Turbulence and Shear Flow Phenomena (TSFP-3).

[38] S. Stolz, N. A. Adams, and L. Kleiser. An approximate deconvolution model for large-eddy simulation with application to incompressible wall-bounded flows. *Physics of Fluids*, 13(4):997–1015, 2001.

[39] Ugo Piomelli and Elias Balaras. WALL-LAYER MODELS FOR LARGE-EDDY SIMULATIONS. *Annual Review of Fluid Mechanics*, 34(1):349–374, 1 2002.

[40] U. Schumann. Subgrid scale model for finite difference simulations of turbulent flows in plane channels and annuli. *Journal of Computational Physics*, 18(4):376–404, 1975.

[41] Sarath Radhakrishnan, Lawrence Adu Gyamfi, Arnau Miró, Bernat Font, Joan Calafell, and Oriol Lehmkuhl. A Data-Driven Wall-Shear Stress Model for LES Using Gradient Boosted Decision Trees. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12761 LNCS:105–121, 2021.

[42] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, New York, NY, USA, 8 2016. ACM.

[43] Peter P. Sullivan, James C. McWilliams, and Chin-Hoh Moeng. A subgrid-scale model for large-eddy simulation of planetary boundary-layer flows. *Boundary-Layer Meteorology*, 71(3):247–276, 11 1994.

[44] Misun Min and Ananias Tomboulides. Simulating Atmospheric Boundary Layer Turbulence with Nek5000/RS. Technical report, Argonne National Labrotory - Mathematics and Computer Science, 2022.

[45] Osborne Reynolds. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical Transactions of the Royal Society of London. (A.)*, 186:123–164, 12 1895.

[46] A. J. Favre. Review on Space-Time Correlations in Turbulent Fluids. *Journal of Applied Mechanics*, 32(2):241–257, 6 1965.

[47] Sergio Hoyas and Javier Jiménez. Scaling of the velocity fluctuations in turbulent channels up to Reτ=2003. *Physics of Fluids*, 18(1), 1 2006.

[48] Christophe Geuzaine and Jean François Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 2009.

[49] Tony Zahtila, Wilson Lu, Leon Chan, and Andrew Ooi. A systematic study of the grid requirements for a spectral element method solver. *Computers & Fluids*, 251(January 2022):105745, 1 2023.

[50] Javier Jiménez and Alfredo Pinelli. The autonomous cycle of near-wall turbulence. *Journal of Fluid Mechanics*, 389:335–359, 1999.

[51] Tamer A. Zaki. From Streaks to Spots and on to Turbulence: Exploring the Dynamics of Boundary Layer Transition. *Flow, Turbulence and Combustion*, 91(3):451–473, 10 2013.

[52] Ronald J. Adrian. Hairpin vortex organization in wall turbulencea. *Physics of Fluids*, 19(4), 2007.

[53] P Fischer, M Min, T Rathnayake, S Dutta, T Kolev, V Dobrev, J.S. Camier, M Kronbichler, T Warburton, K Swirydowics, and J Brown. Scalability of high-performance pde solvers. *International Journal of High Performance Computing Applications*, 34:562–586, 2020.

[54] Misun Min, Michael Brazell, Ananias Tomboulides, Matthew Churchfield, Paul Fischer, and Michael Sprague. Towards exascale for wind enery simulations. revision, 2023.

[55] S. Patel, P. Fischer, M. Min, and A. Tomboulides. A characteristic-based, spectral element method for moving-domain problems. *J. Sci. Comp.*, 79:564–592, 2019.

[56] G.K. El Khoury, P. Schlatter, A. Noorani, P.F. Fischer, G. Brethouwer, and A.V. Johansson. Direct numerical simulation of turbulent pipe flow at moderately high Reynolds numbers. *Flow, Turbulence, and Combustion*, 91:475–495, 2013.

[57] Azad Noorani, Adam Peplinski, and Philipp Schlatter. Informal introduction to program structure of spectral interpolation in nek5000, 2015.