Ref. Ares(2025)10271807 - 25/11/2025

D2.3 Description of multi-fidelity approach to combine RANS and LES simulations

Date: Oct. 31, 2025

Funded by the European Union

This Project has received funding from the European Union's HORIZON Research and Innovation Programme under Grant Agreement number 101096698. The opinions expressed herein reflect the authors' views only. Under no circumstances shall the Horizon Europe program be responsible for any use that may be made of the information contained herein.

Document Identification			
Status	Final	**Due Date**	31/10/2025
Version	1.0	**Submission Date**	25/11/2025

Related WP	WP2	**Document Reference**	D2.3
Related Deliverable(s)	Insert Related Deliverables	**Dissemination Level (*)**	PU
Lead Participant	TUD	**Lead Author**	Akshay Patil, TUD
Contributors		**Reviewers**	Clara Garcia-Sanchez, TUD
			Gerardo Zampino, KTH

Keywords:
Computational Geometry, Synthetic Eddy Method, High-Fidelity Simulation

.

Document Information

List of Contributors	
Name	**Partner**
Akshay Patil	TUD
Clara Garcia-Sanchez	TUD

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	23/10/2025	Akshay Patil	Initial Draft
0.2	31/10/2025	Akshay Patil	Pre-submission draft
0.3	15/11/2025	Akshay Patil	Implementation of the reviewer’s comments
0.4	20/11/2025	Akshay Patil	Implementation of the reviewer’s comments
1.0	25/11/2025	Gerardo Zampino	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Deliverable leader	Clara Garcia-Sanchez (TUD)	20/11/2025
Quality manager	Gerardo Zampino (KTH)	25/11/2025
Project Coordinator	Artem Kulachenko (KTH)	25/11/2025

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation/acronym	Description
CaNS	Canonical Navier-Stokes
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
FFT	Fast Fourier Transformation
GenSDF	Signed-distance-field generator for Cartesian grid solvers
GPU	Graphics Processing Units
HPC	High-Performance Computing
IBM	Immersed Boundary Method
iSTFG	Isotropic Synthetic Turbulence Field Generator
LES	Large-Eddy Simulation
MPI	Message-Passing Interface
SDF	Signed-Distance-Field
uDALES	Urban Dutch Atmospheric Large Eddy Simulation
WP	Work Package

Executive Summary

Wind flow around complex, heterogeneous urban environments are inherently challenging to predict without employing resource-intensive, scale-resolving simulation frameworks, which can constrain the efficiency of wind forecasting. Lower-fidelity approaches, such as Reynolds-Averaged Navier–Stokes (RANS) models, provide a computationally efficient alternative, enabling the incorporation of many simulations to account for boundary conditions and intrinsic uncertainties. However, they are limited by epistemic uncertainty in representing small-scale turbulence. To address this, engineering wind prediction methodologies are increasingly adopting multi-fidelity strategies that combine low- and high-fidelity models, so-called multi-fidelity frameworks, thereby reducing both aleatory and epistemic uncertainties while leveraging high-resolution insights at a manageable computational cost in terms of CPU hours and CO_2 emissions.

Our multi-fidelity framework benefits from the large directionality of our low-fidelity simulations while enhancing the turbulence effects within our results through multiple sets of high-fidelity simulations. The outcomes will not only benefit stakeholders interested on decision making, but it will also support further understanding of airflow dynamics in the built environment. The specific results from our approach are more robust predictions within the urban environment, which can facilitate more informed decisions regarding urban planning and comfort, but as well urban air mobility safety and planning. In addition, fully understanding the airflow effects within urban areas can serve to determine potential locations for vertiports where risks are minimized.

Building upon the developments from WP2.2, a multi-fidelity framework was established using uDALES, a Cartesian LES solver, as the high-fidelity counterpart to OpenFOAM-based RANS solutions, enhancing the predictive accuracy of the unified modelling system. A scalable, memory-efficient signed-distance-field (SDF) generator was implemented to introduce complex obstacles within Cartesian grid solvers without compromising accuracy or computational efficiency. In parallel, a synthetic turbulence generator was developed to accelerate convergence toward statistically stationary flow states in channel-like domains by generating initial conditions that achieve the desired turbulence statistics more rapidly.

These advancements culminated in a Kriging-based multi-fidelity computational framework that effectively couples low- and high-fidelity simulations, enabling robust and efficient wind flow predictions in complex urban environments.

It is worth mentioning the impact of the present task on policy makers responsible for setting regulations related to air quality and the safety of inhabitants in large urban areas. The model provides a series of numerical analyses aimed at identifying the most critical aspects of urban ventilation within streets and city districts. In this context, urban engineers and domain experts benefit from CFD simulations with higher predictive power to assess vorticity, flow velocity fields, and turbulent kinetic energy in urban environments. The influence of building positions and geometries is also a key factor that can be incorporated and evaluated within the modelling framework.

Another important aspect concerns the reduction of computational load, which results in lower server usage and reduced operational costs, enabling more accessible and scalable cloud-based CFD services. This, in turn, leads to reduced CO_2 emissions due to the lower number of CPU hours required for high-fidelity analyses.

2 Introduction

Scale-resolving turbulent flow simulations around complex objects have become increasingly accessible for engineering problems due to the increasing computational resources [1-4]. While many methods have been introduced for complex objects within the flow, such as body-conforming grids, as shown in Figure 1 [3, 5], the use of the immersed boundary method (IBM) has become increasingly popular due to its efficient underlying algorithm on regular grids [6, 7]. Since IBM does not require the grid to conform to the object, the underlying grid data structure can leverage the Cartesian grid and use efficient pressure solvers for the governing equations [8]. For complex objects, it becomes essential to accurately locate the boundary of the immersed object on the computational lattice, which can be achieved using the signed-distance-field (SDF).

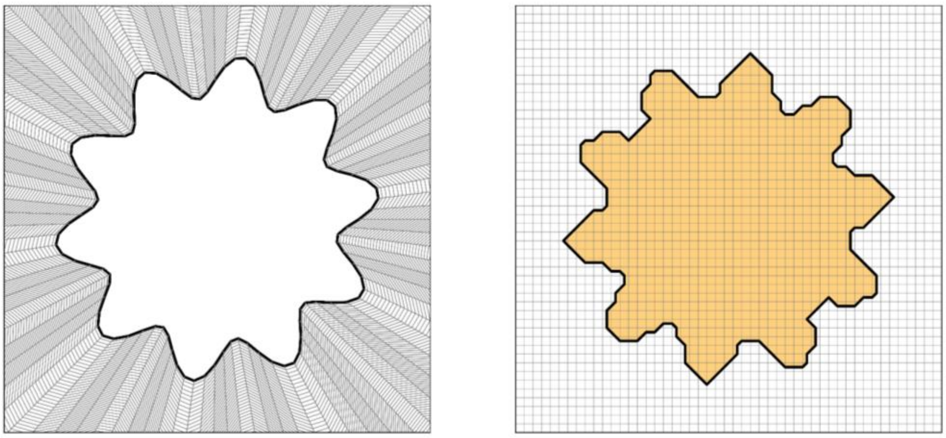

Figure 1: (Left) Body conforming mesh used in typical computational fluid dynamics applications, (Right) Immersed Boundary Method over a regular Cartesian grid.

While SDF generators are not a novelty [9], the closed-source nature of existing solutions has limited their accessibility for CFD applications. Additionally, as the problem size for scale-resolving simulations increases, the memory requirements for generating the SDF can grow drastically, necessitating a distributed-memory framework to efficiently parallelize the SDF generation. In this work, we employ the distributed-memory paradigm enabled by the Message Passing Interface (MPI) to overcome this limitation, allowing for a scalable algorithm that extends beyond billions of grid points and corresponding surface triangulations.

In most cases involving realistic urban configurations, it is often difficult to achieve stationary flow conditions independently using periodic configurations, as they do not accurately represent the wind conditions typically observed in situ. Consequently, a precursor simulation is typically used to supply the inflow boundary conditions. To this end, most precursor simulations are run with periodic channel flow-like configurations that can be efficiently executed without incurring significant costs. These simulations can then be further coupled at the inlet of the driven domain, where investigations of the urban area are carried out. Turbulent channel flows have significantly improved the overall understanding of wall-bounded turbulence, as detailed in the extensive body of literature [18-26]. While such studies provide a fundamentally novel insight into flow physics, the computational resources required to simulate such flows can become increasingly demanding as the flow Reynolds number increases. Recent estimates by Horwitz [26] suggest that a turbulent channel flow simulated by Vela-Martín et al. [27] using 512 Graphics Processing Units (GPU) can use up to 5.98 × 105 kilo-Watt-hours (kWh) of energy and emit an equivalent of 6894 kg of CO2 when running over a course of 2734 wall-clock hours. The simulation by Vela-Martín et al. [27] solved for the three-dimensional flow field at a friction Reynolds number $Re_\tau \equiv u_\tau H/\nu$ = 5303, where $u_\tau$is the friction velocity, H is the channel half-height, and ν is the kinematic viscosity of the fluid. As illustrated by this study, achieving statistically stationary flow conditions can pose a significant hurdle, which requires long simulation spin-up times before flow parameters can be averaged to gain generalizable insights.

Without appropriate initial conditions for the channel flow simulations, most simulation frameworks generate three-dimensional flow fields initialised using simple analytical profiles for the streamwise velocity components with white noise [28,29]. For established datasets, such as the Johns Hopkins turbulence dataset maintained by Graham et al. [29], a similar simulation spin-up is also mentioned. The flow is run with a constant bulk velocity forcing, i.e., momentum forcing, like Nelson and Fringer [28], to obtain the initial conditions. In some studies, a pair of counter-rotating vortices [30] is added to the linear log-law profile to trigger the transition to turbulence and accelerate momentum mixing in the vertical direction, thereby reducing the CPU time to achieve statistically stationary flow conditions [29]. However, as discussed by Nelson and Fringer [28] and Costa [8], the simulations require over 10 turnovers, $T_\epsilon \equiv H/u_\tau$, periods to reach stationary flow conditions. As the flow Reynolds number increases, the computational cost can grow dramatically, limiting the resources available for collecting valuable statistical data. To bridge this gap, we developed a scalable isotropic synthetic turbulence field generator (iSTFG) capable of synthesising initial conditions for arbitrary channel flows based on user-provided input data. In this study, we used iSTFG-generated initial conditions for two Reynolds-number-based, pressure-driven channel flows to investigate how the spin-up time compares with existing methods for achieving stationary flow conditions in prototypical channel domains. Additionally, we examined the impact on the convergence of flow statistics as a function of computational domain size, model parameters (i.e., the isotropic integral length scale), and input data mismatch. The focus was on understanding the effect of time-marching a given initial condition without any additional forcing, apart from the constant pressure gradient driving the flow.

Finally, by leveraging a combination of these pre-processing methodologies, we propose a multi-fidelity computational framework that integrates low-fidelity RANS models with a hierarchy of coarse-to-fine-resolution Large-Eddy Simulation (LES) methods. This helps to more accurately predict wind-related risks associated with urban mobility and pedestrian comfort.

2.1 Purpose of the document

As discussed in the motivation, to reduce pre-processing time and overall computational cost without compromising prediction accuracy, we employ a multi-fidelity approach to enhance predictive capabilities. While the low-fidelity results provide large parameter sweeps, the high-fidelity results incorporate unsteady and small-scale effects that are otherwise not possible to infer using the low-fidelity results. In this document, we have detailed how the pre-processing utilities can benefit the simulation frameworks and laid out a concise plan for the multi-fidelity framework.

2.2 Relation to other project work

Building on the low-fidelity framework, this work package develops a foundational framework using an immersed boundary large-eddy simulation technique that solves part of the small-scale physics, which are entirely modelled in the low-fidelity framework, to better understand the flow within the urban canopy. The unsteady nature of the simulations in the higher-fidelity framework can be leveraged to design more effective deep learning models that predict UAV trajectories, such as those for WP3 and WP5. The connection between the WP is performed by means of shared data in the form of velocity vectors and turbulence levels within the urban environment. These fields provide essential information that can support determining which flight paths minimize risk for UAV vehicles depending on their size and resistance characteristics. The low/high-fidelity fields obtained through CFD simulations are furnished to WP2 and WP4 to first optimize with deep learning UAV trajectories (WP2) and then plan and control UAV paths within the integration WP (WP5).

2.3 Structure of the document

Building on the low-fidelity framework, this work package develops a foundational framework using an immersed boundary large-eddy simulation technique that replicates the small-scale physics, which is entirely modelled in the low-fidelity framework, to better understand the flow within the urban canopy. The unsteady nature of the simulations in the higher-fidelity framework can be leveraged to design more effective deep learning models that predict UAV trajectories, such as those for WP2 to whose this task belongs, and WP4. In this context, the present task will directly support the development of noise prediction models. Since noise emission and propagation are strongly influenced by the characteristics of the velocity field, accurately resolving the local flow dynamics is essential. In particular, both the magnitude and direction of the velocity field determine how sound waves are advected, refracted, or attenuated as they travel through the urban environment. As a result, flow features such as shear layers and recirculation zones can improve or suppress the noise propagation in specific directions.

Chapter 3: Signed-distance-field generator for Cartesian grid solvers,

Chapter 4: Synthetic turbulence field generator for statistically stationary flow conditions,

Chapter 5: Description of the multi-fidelity framework and preliminary results for Delft University of Technology campus,

Chapter 6: Concluding remarks.

3 Signed-Distance-Field Generator for Cartesian Grid Solvers

The GenSDF[1] software is a general-purpose code developed to accurately and efficiently obtain the SDF over a Cartesian grid for an arbitrary triangulated geometry.

3.1 Software Architecture

The GenSDF software is written in Fortran 2003 standard and later [10], providing a simple yet computationally efficient framework for developing and maintaining the code. Distributed-memory parallelism is achieved through the MPI interface (compatible with the MPI 4.0 standard and above), where the computational grid is decomposed linearly along the x-axis into N chunks, with N corresponding to the number of MPI ranks used in the parallel version of the code. Figure 2 presents a flowchart illustrating the main components of the software described in this paper. As outlined in Step 2 of the algorithm, the bounding box coordinates of the triangulated geometry are used to co-locate it on the computational grid. Based on this information, a large positive distance is assigned to grid points outside the bounding box (hereafter referred to as the B-Box in Figure 2), since these points are known a priori to lie outside the geometry. This B-Box methodology enables the efficient exclusion of computationally expensive distance queries for points that have already been identified as being outside the geometry. The B-Box also includes a user-specified stencil width (s_b), where s_b denotes the number of computational grid points in the wall-normal direction of the B-Box. Additionally, s_b is used to define the narrow-band distance calculation region, representing the number of grid points away from the geometric surface over which the distance calculation is performed. After the B-Box is co-located in Step 2, it is decomposed according to the user-specified number of MPI ranks, as described in Step 3. The domain decomposition is performed along the streamwise direction, which is typically the longest dimension in such flow simulations. Once the decomposition is complete, each MPI rank computes the local distance calculations for the surface triangles within its respective coordinate extent. Finally, Step 4 assembles the decomposed domain by managing boundary values and performing file write operations.

[1] The details presented in this chapter are taken from the publication: *Patil, A., Paranjothi, U. C. K., & García-Sánchez, C. (2025). GenSDF: An MPI-Fortran based signed-distance-field generator for computational fluid dynamics applications. SoftwareX, 30, 102117.*

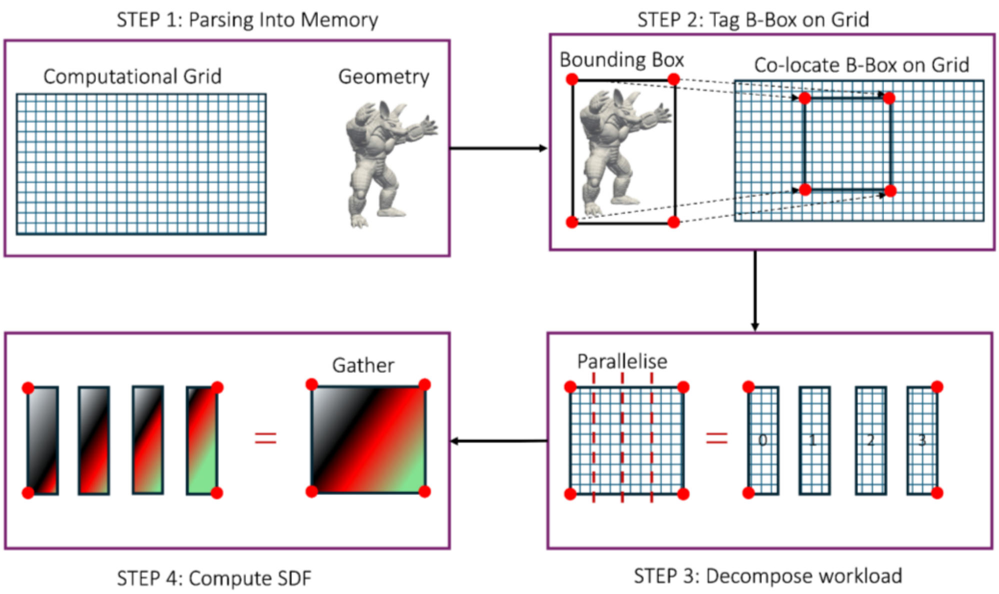

Figure 2: Flowchart of the key steps in the code. In step 3, the numbers represent the MPI rank IDs. The flowchart presents a 2D example of the B-Box; however, the code works in 3D.

3.2 Software Functionalities

The software has three core functionalities:

- Parse OBJ geometry and load it into memory.
- Calculate the distance between a query point and a triangular face.
- Parallelise the workload for efficient and accurate computations.

Within these core functionalities, the software first parses the input file where the user provides information about the input geometry, input grid type, the default distance value for points outside the geometry, and the stencil width. This user input file parser is contained in the subroutine called read_inputfile. Once the input data is known, the computational grid is parsed into memory on each MPI rank using the read_cans_grid subroutine. Then, the surface triangulated geometry is parsed into memory using the read_obj subroutine, which loads the vertex, vertex normal (assumed to be pointing outward), and face data. Once the geometry is loaded into memory, the bounding box corresponding to the geometry is calculated, which is then used to co-locate the extent over which the distance is queried. The computationally intensive workload is housed in the subroutine titled compute_scalar_distance_face. The algorithm to compute the signed distance is designed to scale the computational effort based on the number of vertices in the triangulated geometry. Specifically, in this implementation, the outermost loop iterates over the total number of faces within the triangulated geometry. As presented in Figure 4, each triangular face on the surface geometry is composed of three vertices, where an average surface normal pointing outwards is also defined (not shown in Figure 4). Knowing the x, y, and z coordinates of the vertices, a face-local bounding box can be obtained readily as the computational grid (shown with the blue grid lines) constitutes a simple Cartesian structure. This bounding box encompasses all candidate grid points for which a distance calculation is considered and excludes all other points that are relatively farther away from the triangulated face under consideration. Once the face-local bounding box is known, the distance between the point and the triangular face is calculated using a well-established algorithm, as described in Eberly [11]. The face-local bounding box circumvents the brute-force computational cost, which would otherwise scale as $(N_c\ x\ N_f)$, where N_c is the number of computational grid points and N_f is the total number of triangular faces in the input geometry. The central idea is to reduce the pre-factor N_c by considering only the computational grid points in the immediate vicinity of the requisite face controlled by the s_b input parameter, thus reducing the overall computational cost to $\sim$ approximately $(s_b\ x\ N_f)$. In this framework, $s_b \sim (10)$ while $N_c \sim (10^7)$ thus resulting in a relatively large reduction in the computational cost associated with calculating the SDF when compared to the brute-force method. Figure 3 presents a simple sketch considering s_b = 1 around the face (an edge in 2D) marked in red and the corresponding points considered for calculating the distance query marked in black- and purple-filled circles (closest point to the face). The thick-purple lines connecting the filled-black circles are chosen based on the value of $s_b$prescribed by the user in each lattice direction around the point of consideration in 3D. The pseudo-code of the algorithm is detailed below. A sign-unaware distance is calculated with the method proposed by Eberly [11], which further requires information from the underlying geometry to correctly tag the positive or negative sign associated with the point being inside or outside the geometry. Since the OBJ file format supports storing the vertex normals as shown in Figure 4, the distance can be tagged to each query point by taking the inner product between the average vertex normals corresponding to the face and the segment connecting the query point and the face centre given by

$$L^{tag}{}_{q_i} = sgn(n_f \cdot k_{q_i - o}), \qquad (1)$$

Where $L^{tag}_{q_l}$ is the sign tagged to the distance calculated for the respective query point q_i, *sgn* is the sign function that returns either a 1.0 or −1.0 (double-precision floating point), n_f is the vertex-averaged normal vector associated with the face, and $k_{q_i - o}$ is the vector originating from the query point and pointing towards the face centre. Query points returning a positive sign tagged distance are tagged to be inside the geometry, while the query points that return a negative sign tagged distance are tagged to be outside, as sketched in Figure 3.

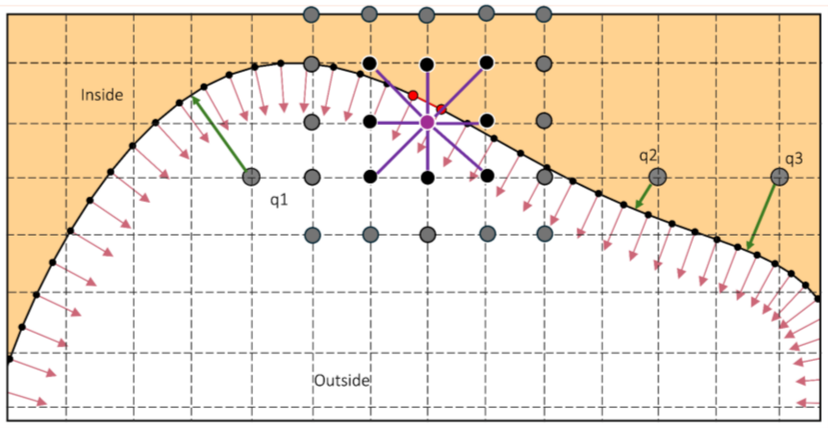

Figure 3: 2D sketch illustrating the sign-tagging logic used in this implementation. The black solid line with black-filled circles represents the faces of the geometry, where each segment corresponds to an individual face. Vertex normals originate at each vertex and point in the outward normal direction. The green arrows, originating from the query points (q_1, q_2, and q_3), indicate directions toward the closest face on the geometry. The black dashed lines denote the computational grid on which the query points are located. The red edge connecting the red-filled circles corresponds to the edge under consideration for the distance query, while the black-filled circles represent the computational grid points for which the distance is computed. The triangulation (i.e., the faces) is shown as isotropic in size; however, this is not a requirement of the algorithm. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Since the user is allowed to specify a stencil width around the geometry where the distance is calculated, the default initialization for the SDF value does not correctly tag the values inside the geometry that should retain a negative value. During the initialisation step, all points are originally tagged to be outside the geometry and assigned a large positive distance value specified by the user in the user input parameters. The effect of varying stencil width without properly handling the internal values is evident in Figure 5, where the algorithm incorrectly assigns a positive default value within the geometry. To avoid this situation, increasing the stencil width can be potentially beneficial; however, the computational cost scales with the stencil width. Therefore, we utilise a simple 3D flood-fill algorithm that assigns a large negative value to internal values bounded by the SDF. This is a critical part of the algorithm as the SDF is used to identify the location of the solid interface (i.e., SDF = 0.0); thus, without the final flood-fill process, the CFD solver could potentially return multiple interface locations. Finally, once the SDF is computed for each individual MPI rank, the gather_array subroutine collects the decomposed parts of the array and stacks them into a single contiguous array, which is then written out to a binary file format for further use within the CFD solver.

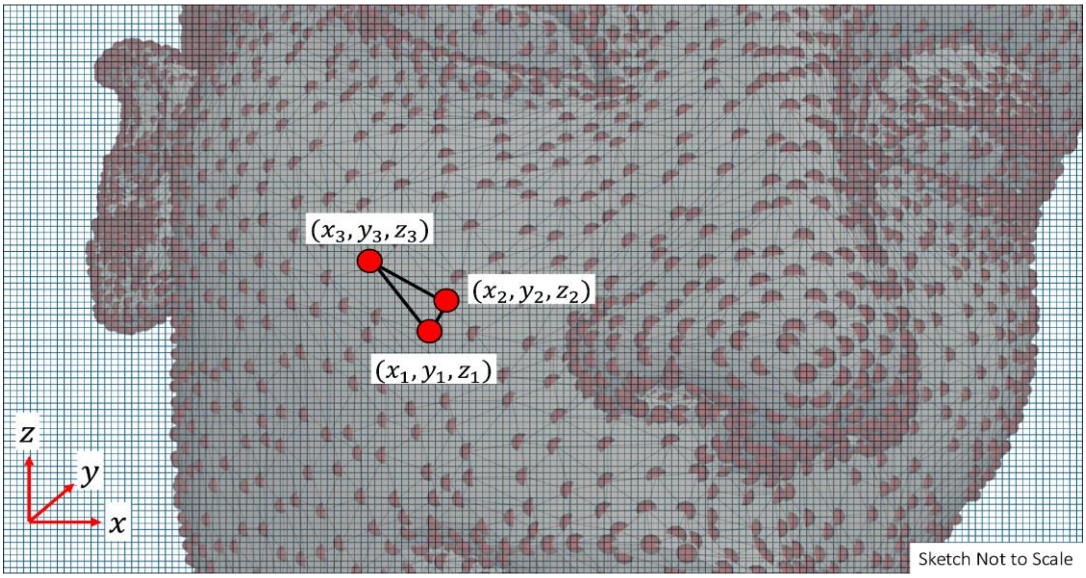

Figure 4: Example sketch of the triangulated surface geometry overlaid onto the computational grid (2D example). The three filled red circles correspond to the vertices of the triangular face denoted by the solid black lines.

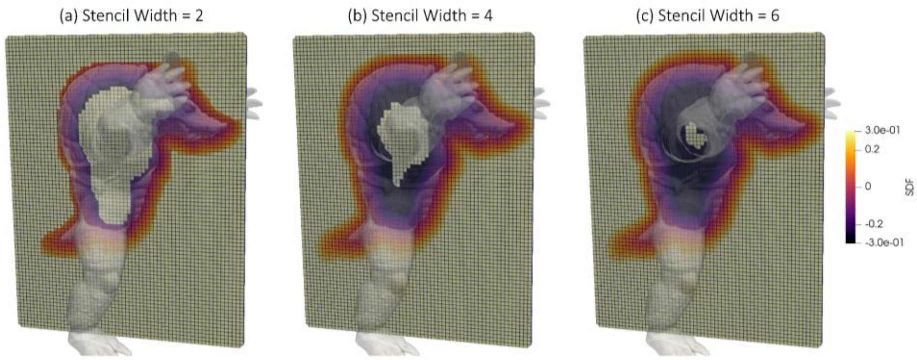

Figure 5: Example case illustrating the effect of stencil width when the flood-fill algorithm is not used post-signed-distance calculation. The size of the internal region with a default positive value decreases as the stencil width increases.

This approach not only localizes the distance query but also allows for an efficient parallelization strategy, as no global communication is required when such a distance query is requested. Since the outer loop iterates over the total number of faces in the triangulated surface geometry, certain limitations must be considered for this algorithm. Geometries with long and slender triangles may render a degenerate SDF. This is especially true for geometries that comprise a large collection of blocks, such as cubes (e.g., buildings). Additionally, for cases where the underlying computational grid is relatively finer compared to the surface triangulation, there may be instances where certain computational points result in a degenerate SDF. A simple fix for both scenarios is to refine the surface triangulation using a meshing tool or the instructions provided within the published repository.

Algorithm 1 Compute the Minimum Distance Between a Point and a Triangle

Require: Point P, triangle $\triangle ABC$ with vertices A, B, C
Ensure: Minimum distance d between P and $\triangle ABC$

Compute edge vectors:
$E_0 \leftarrow B - A$
$E_1 \leftarrow C - A$
$V \leftarrow P - A$
Compute dot products:
$d_{00} \leftarrow E_0 \cdot E_0$
$d_{01} \leftarrow E_0 \cdot E_1$
$d_{11} \leftarrow E_1 \cdot E_1$
$d_{20} \leftarrow V \cdot E_0$
$d_{21} \leftarrow V \cdot E_1$
Compute barycentric coordinates:
$\text{denom} \leftarrow d_{00} d_{11} - d_{01}^2$
$v \leftarrow \frac{d_{11} d_{20} - d_{01} d_{21}}{\text{denom}}$
$w \leftarrow \frac{d_{00} d_{21} - d_{01} d_{20}}{\text{denom}}$
$u \leftarrow 1 - v - w$
if $u \geq 0 \wedge v \geq 0 \wedge w \geq 0$ **then**
    $P_c \leftarrow A + vE_0 + wE_1$
    return $d \leftarrow \|P - P_c\|$
end if
Check closest vertex cases:
if $v \leq 0 \wedge w \leq 0$ **then**
    return $d \leftarrow \|P - A\|$
else if $u \leq 0 \wedge v \leq 0$ **then**
    return $d \leftarrow \|P - C\|$
else if $u \leq 0 \wedge w \leq 0$ **then**
    return $d \leftarrow \|P - B\|$
end if
Check closest edge cases:
if $u \leq 0$ **then**
    $P_c \leftarrow \text{ClosestPointOnSegment}(P, B, C)$
else if $v \leq 0$ **then**
    $P_c \leftarrow \text{ClosestPointOnSegment}(P, C, A)$
else if $w \leq 0$ **then**
    $P_c \leftarrow \text{ClosestPointOnSegment}(P, A, B)$
end if
return $d \leftarrow \|P - P_c\|$

Algorithm 2 Closest Point on a Line Segment

Require: Point P, segment endpoints X, Y
Ensure: Closest point P_c on segment XY
1: **Compute projection parameter:**
2: $t \leftarrow \frac{(P-X)\cdot(Y-X)}{(Y-X)\cdot(Y-X)}$
3: **Clamp** t **to** $[0, 1]$:
4: $t \leftarrow \max(0, \min(1, t))$
5: **Compute closest point:**
6: $P_c \leftarrow X + t(Y - X)$
7: **return** P_c

3.3 Input requirements and performance

GenSDF is designed with consideration for the CFD end-user, who requires input parameters and data to work as detailed in this paper. Some of these requirements are:

- Geometry. The geometry must consist of triangles and is manifold and watertight. In cases where the computational grid spacing is larger than the triangulation of the geometry, water tightness is not a strict requirement. The geometry must be stored in the wavefront OBJ file format, as it stores the vertex normal information that is central to the implementation considered in this study. Additionally, this file format is supported and used in many open-source software applications, where the input geometry is used for setting up scale-resolving simulations [12,13]. The algorithm ensures that the coordinate axes x_i are aligned such that i = 1, 2, 3 correspond to the streamwise, spanwise, and vertical directions, respectively. The bounding box of the input geometry must be contained within the bounding box of the computational geometry, either completely or partially. In case the input geometry is completely outside the bounds, the SDF will simply return a default value for all computational grid points.
- Computational grid. The computational grid over which the SDF is to be calculated needs to be provided. The code, in its current form, is directly compatible with one of the widely used Canonical Navier-Stokes (CaNS) models developed by Costa [8]. However, since most scalable CFD solvers that would require the use of GenSDF have a similar structure, i.e., isotropic grid spacing in two directions, and non-isotropic grid spacing in one direction, the grid can be ported to CaNS format. Specifically, the CaNS grid structure

consists of two files, namely, a. geometry.out, and b. grid.out. These two files are assumed to be hosted in the data directory relative to the location of the executable. The geometry.out file contains two lines with three columns of information. The first line contains the total number of grid points, and the second line contains the length of the domain with respect to the computational grid origin (default origin: 0, 0, 0) in the streamwise, spanwise, and vertical directions, respectively. The grid.out file contains five columns and nz lines, where nz is the number of grid points in the vertical direction. The first column is typically empty and has a dummy value of 0.0, while the second and third columns correspond to the cell centre and cell face locations in the vertical directions, respectively. The fourth and fifth columns correspond to the grid spacing for the cell centre and the cell face locations. Both the geometry.out and grid.out columns are separated by a space and do not contain a file header.

- Parameters. A complete list of input parameters required in addition to the input geometry is detailed in Listing 1. Each line preceded by a '!' corresponds to a comment. The scalar value refers to the initial value assigned to all points assumed to be outside the geometry. The real and int keywords correspond to floating-point and integer data types, respectively. The non-uniform-grid boolean flag can either be set to 'T' or 'F' for True and False, respectively.

Listing 1: Input parameters required for the code with the default file name *parameters.in*

```
! Name and location of the inputfile
'data/armadillo_withnormals.obj'
! Scalar value (real), stencil width (int),
    progressbarsize (int)
100.0 4 10
! nx, ny, nz (Computational Grid)
512 128 128
! r0 (Origin of the computational grid)
0.0 0.0 0.0
! non_uniform_grid
F
```

To evaluate the accuracy and the performance of the software, we first consider the effect of increasing resolution for a sphere with a radius of r_c = 0.05 units centred at (0.5, 0.5, 0.5). The background computational grid is discretized using three grid resolutions $r_c/\Delta = [12.8, 25.6, 51.2]$. Figure 6(a) compares the results using GenSDF, where the level set (SDF = 0.0) is shown alongside the analytical geometry. With increasing grid resolution, the level set converges toward the analytical geometry. For relatively complex geometry, the level set accurately captures the underlying geometry, even at a relatively coarse resolution, as shown using the orange solid line in Figure 6(b–d). To demonstrate the performance of the algorithm, we consider three cases, namely the Orion [14], the Stanford Dragon, and Lucy [15], with two grid sizes with a resolution of $\Delta = [1/512, 1/1024]$. The original geometry was re-scaled to fit a bounding box (of the geometry) with size $x_{min} = (0.5, 0.5, 0.5)$, $x_{max} = (1.0, 1.0, 1.0)$ within the cube (background computational grid) of side length 1.5 units. Table 1 lists the time required to generate the SDF using eight cores on a single-socket CPU. It is clear to see that, despite the relatively simple computational structure, the code is fast and accurate for complex geometries.

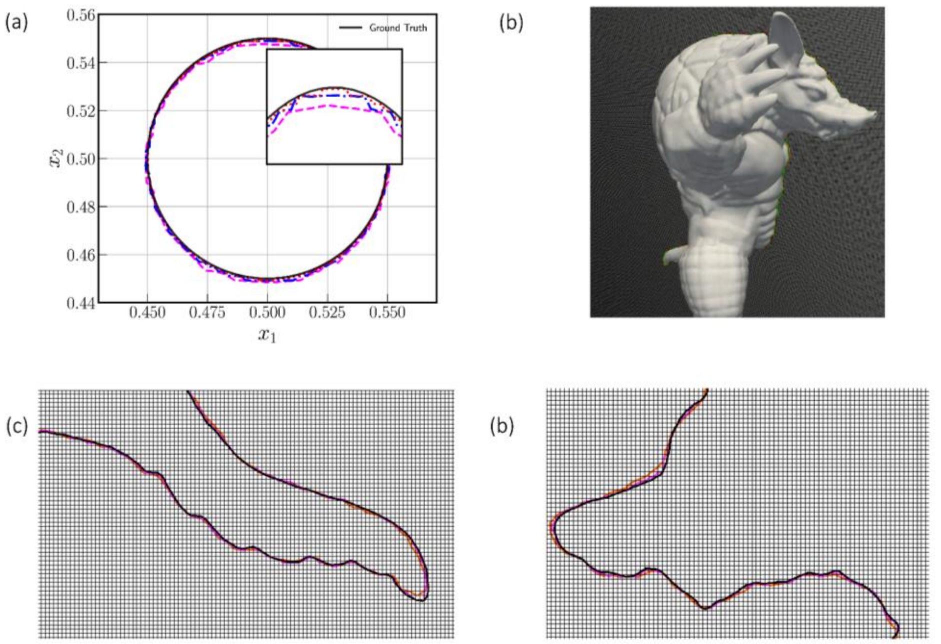

Figure 6: (a) SDF level-set comparison against analytical geometry for varying computational grid resolution. The dashed-magenta, dash–dot-blue, and dotted-red lines correspond to $r_c/\Delta = [12.8, 25.6, 51.2]$, respectively, while the solid black line represents the ground truth. (b–d) Comparison of the level-set against a relatively complex geometry with an effective resolution of $\Delta = [1/512, 1/1024]$ marked using orange and magenta solid lines, respectively, while the solid black line represents the exact geometry. The background grid corresponds to a resolution of $\Delta = 1/1024$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

Table 1: Computational time required for the three geometries tested using GenSDF as a function of the number of faces in the geometry for constant background grid resolution.

Geometry Name	N_f	$\Delta = 1.0/512$	$\Delta = 1.0/1024$
Orion	51736	1.64s	2.46s
Stanford Lucy	99970	1.63s	1.55s
Stanford Dragon	249882	4.90s	5.21s

3.4 Illustrative Examples

Despite some shortcomings mentioned above, GenSDF has been successfully applied to extreme-scale simulation setups, as illustrated in the following section. By default, the code assumes that the grid is staggered such that the locations of u, v, w (velocities in streamwise, spanwise, and vertical directions, respectively) and p (pressure and any other cell-centre parameters of interest, i.e., scalars) are different, thus generating four different arrays corresponding to the location of the variables. The code also provides an indication of the expected minimum memory required to run the specific program based on the total number of arrays initialised by the software. This is important when the problem size is large and the user requires a minimal memory estimate to schedule the SDF generation on HPCs.

Figs. 7, 8, and 9 depict the suitability of GenSDF for multi-scale and complex geometries. Figure 7 depicts the SDF generated for an artificially generated coral reef containing a total of 14.85 million vertices and 29.75 million faces on a background non-isotropic computational grid containing 1.2 billion grid points. The Stanford dragon exhibits relatively sharp curvature and slender cross-sections as seen in Figure 8, and GenSDF can sufficiently capture the details without introducing a large computational overhead (see Table 1). A relatively extreme-scale case is depicted in Figure 9, with 2 billion grid points and 25 million triangulation faces. The grey colour marks the level set of 0, representing the solid boundary of the buildings, while the colour indicates the distance field.

.

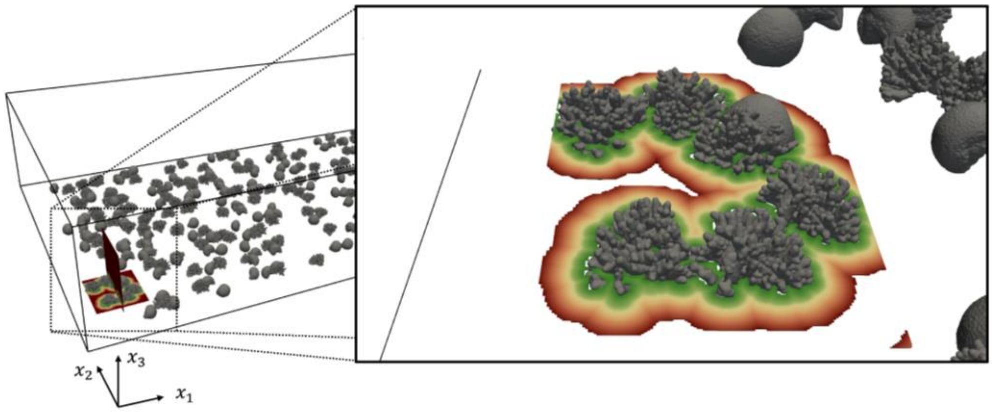

Figure 7: SDF generated for a multi-scale artificially generated coral reef from [17].

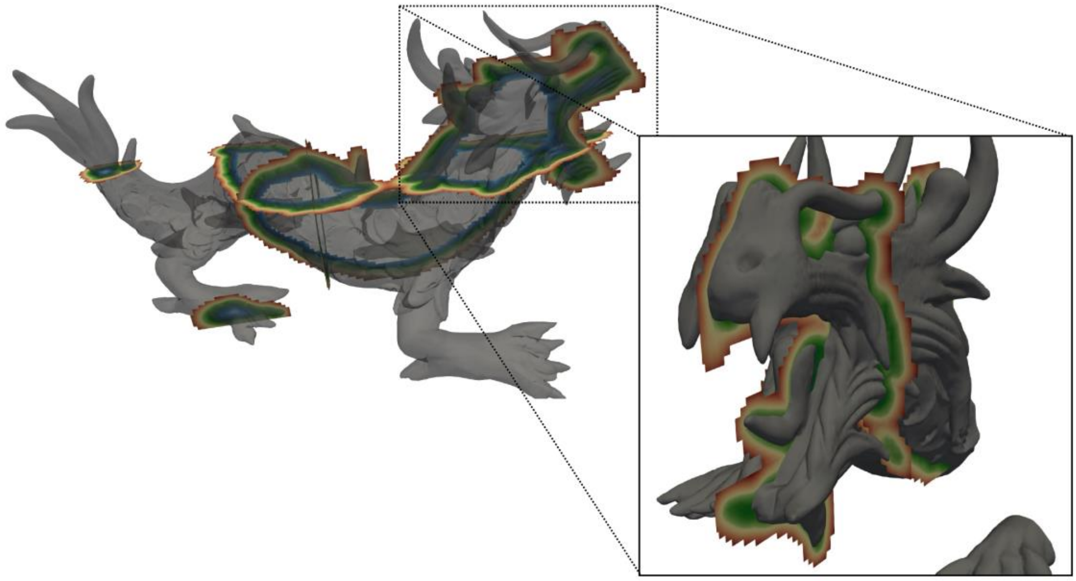

Figure 8: SDF generated for complex geometry corresponding to the Stanford Dragon [15].

Figure 9: SDF generated for the Delft University of Technology central campus with a background grid size of 2 billion grid points and 25 million triangulation faces for the underlying geometry. Only a section of the grid is shown in the figure.

GenSDF offers an open-source, scalable, and accurate method for generating a signed distance field for complex objects over a Cartesian grid with variable vertical grid spacing. The development of this code was primarily motivated by the need for a scalable and distributed memory solution to generate a signed distance field for computational fluid dynamics applications. Some advantages of GenSDF are: (1) Low memory footprint, (2) Scales over 100s of CPUs, (3) Easily portable to different CFD solvers. The software also provides detailed documentation of the source code and example usage. GenSDF is continually updated and tracked through the GitHub repository, where users can directly contribute.

4. Synthetic Turbulence Field Generator for Statistically Stationary Flow Conditions[2]

In this study, we consider the non-dimensional incompressible Navier–Stokes momentum equations given by

$$\partial_t u_i + \partial_j u_j u_i = -\partial_i p + Re^{-1} \partial_j \partial_j u_i + \Pi_c \delta_{i1} \qquad (1)$$

subject to the incompressible continuity equation given by

$$\partial_i u_i = 0 \quad (2)$$

Where t is time, u_i is the velocity vector, p is pressure, $Re_\tau \equiv u_\tau H/\nu$, Π_c is the driving pressure gradient, and δ_{ij} is the Kronecker delta function. The $(\cdot)^*$ notation represents the non-dimensional parameters obtained using u_τ and H. Using this choice of non-dimensionalisation, the driving pressure gradient is unitary. The governing equations are numerically integrated using the open-source massively parallel CaNS solver developed by Costa [8]. CaNS solves the governing equations using a second-order accurate spatial discretisation and a third-order accurate temporal discretisation, employing the low-storage Runge–Kutta 3-step method with the fractional step algorithm [32]. The flow variables are arranged on a staggered grid where scalars are placed at the cell centre while vector components are located at the cell faces [33]. CaNS has been extensively validated for channel flow simulations; further details can be found in Costa [8] and will not be discussed here for brevity.

For the medium-size domain, the channel has dimensions $L_{x_1}\, x\, L_{x_2}\, x\, L_{x_3} = 4\pi H\, x\, 2\pi\, x\, H$, where x_i corresponds to the coordinate axes in streamwise, spanwise, and vertical directions, respectively. The flow is driven by a constant pressure gradient Π_c = 1.0 subject to periodic boundary conditions in the streamwise and spanwise directions, and a no-slip boundary condition at x_3 = 0 and free-slip boundary where $\partial_3 u_1 = \partial_3 u_2 = 0$ and u_3 = 0 at $x_3 = H$. The flow field is initialised using three different initial conditions: namely, inverse linear profile superposed with white noise and a pair of counter-rotating vortices (hereafter referred to as linear profile), linear log-law profile superposed with white noise and a pair of counter-rotating vortices (hereafter referred to as log profile), and synthetically generated three-dimensional flow field using the iSTFG based on the work by Kim et al. [34] (hereafter referred to as synthetic profile).

---

[2] The details presented in this chapter are taken from the publication: *Patil, A. and García-Sánchez, C., (2025), Fake it Till You Make it: Synthetic Turbulence to Achieve Swift Converged Turbulence Statistics in a Pressure-Driven Channel Flow, Computers & Fluids, 106733.*

The linear profile initial condition is inspired by Nelson and Fringer [28], and is given by

$$u_i = 2U_o\,(1 - x_3/H)\delta_{i1} + P(-\alpha U_b, \alpha U_b) \qquad (3)$$

Where P is sampled from a uniform random distribution, α is the white noise amplitude parameter set to a value of 0.7 as recommended by Nelson and Fringer [28] for all cases. The scaled bulk velocity for the linear profile is given by Nelson and Fringer [28]

$$U_o = u_\tau/\kappa\,[\,log(H/z_o)\,+\,z_o/H - 1] \qquad (4)$$

where $z_0 = \upsilon/(9u_\tau)$. An inverse linear profile is used to effectively trigger the transition to turbulence as the shear stress at the bottom wall is large during the first eddy turnover. The plane-averaged shear stress is known a priori for the analytical cases. The log velocity profile is only applied to the streamwise velocity component, like equation (4),

$$u_1 = u_\tau/\kappa\,log(x_3 u_\tau/\nu) + 5.5 \qquad (5)$$

While the pair of counter-rotating vortices is initialised by prescribing the spanwise and vertical velocity components, as described by Henningson and Kim [30], the primary utility of adding this descending pair of vortices is to trigger a transition for the analytical profiles. As detailed in Nelson and Fringer [28], the linear and the log profiles may not always transition to a turbulent flow state. However, adding multiple pairs of vortices may not always lead to better performance, as it can introduce larger TKE levels during the initial transient that can take a long time to dissipate as the flow evolves, thus requiring a relatively longer spin-up. Consequently, for all the cases discussed in this work that make use of the pair of vortices, only a single pair was introduced. The bulk dimensional velocity U_b is derived from Pope [34] with an additional scaling factor of 0.5 to avoid overshoot, given by

$$U_b = 0.5\,[\nu/H]\,[Re_\tau/0.09]^{1/0.88} \qquad (6)$$

The choice for the two methods is detailed in Eqs. (3) and (5) are motivated by Nelson and Fringer [28]; Costa [8].

Figure 10 depicts the initial condition generated using the iSTFG, while Figure 11 shows the plane-averaged profiles for the mean velocity and the root-mean-squared (rms) and $\langle u'_1 u'_3\rangle$ components compared against the data from Moser et al. [20]. The rescaling from $Re_\tau = 590$ to $Re_\tau = 500$ is done through a transformation between the non-dimensional profiles from the reference dataset. Specifically, knowing the values for ν^t, H^t, and $u^t{}_\tau$ for the target simulation, the dimensional profiles are first interpolated from the reference dataset onto the target simulation grid in the vertical direction. When interpolating the profiles on the target simulation grid, the vertical coordinate must be normalised as $x^*{}_3 = x_3/L_{ref}$, and all the other terms are non-dimensionalised using the friction velocity for inner scaled parameters. The rescaling works best when the difference between the target and reference datasets is not significant. However, for target simulations where a reference dataset is not available, the closest available dataset can provide a reasonable starting point for spinning up the simulation. As shown in Figure 11, the plane-averaged profiles agree well with the reference data, demonstrating the suitability of this method for generating initial conditions in channel flow simulations. The novelty of this work does not lie in the iSTFG itself, but in its application for generating initial conditions for channel flows.

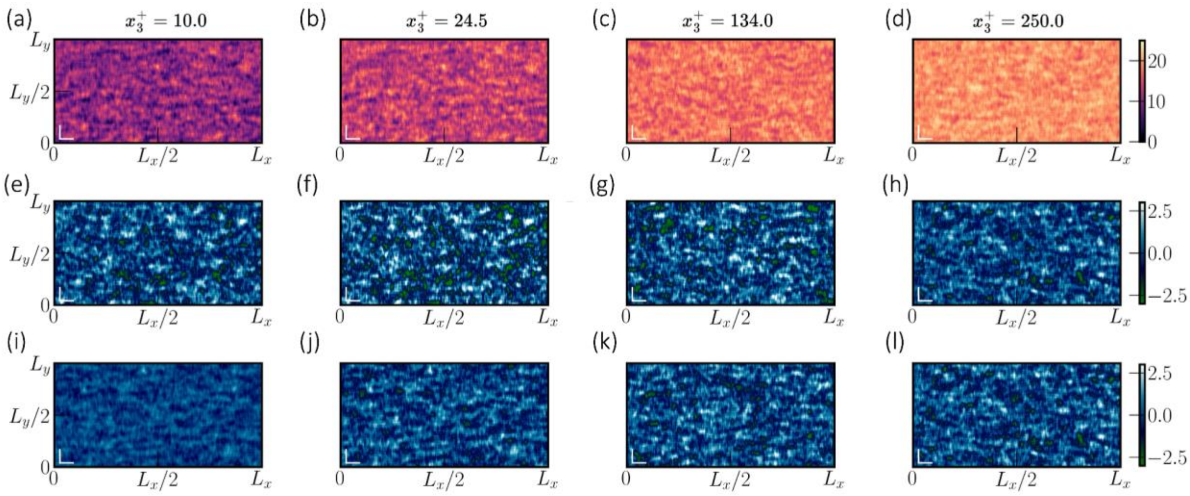

Figure 10: Snapshot of the initial condition generated using the iSTFG at various x_3^+ locations above the wall for $Re_\tau = 500$. The rows from top to bottom correspond to the streamwise, spanwise, and vertical velocities, respectively. Two white lines at the bottom left corner of each panel provide a reference length scale of 400 wall units in the vertical and horizontal directions. The velocity marked using the colour maps is non-dimensionalised using the friction velocity.

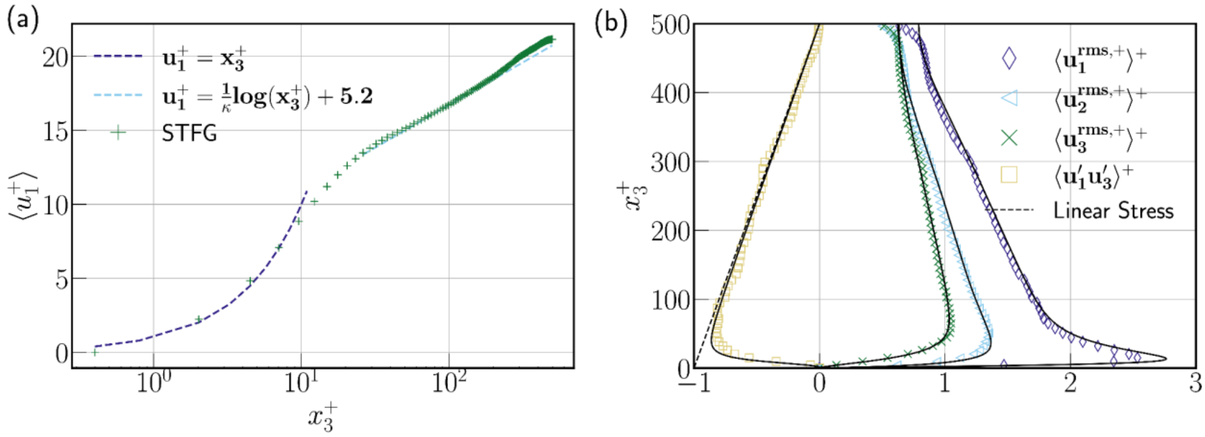

Figure 11: (a) plane-averaged streamwise velocity profile comparison of the iSTFG data against linear, log-law profiles. (b) Root-mean-squared velocity and Reynolds stress profiles compared against the reference data from Moser et al. [20]. On both panels, data markers are shown for every fourth data point.

Table 2 presents the various simulations conducted in this paper. All simulations were run on the DelftBlue (DB) high-performance computing centre at Delft University of Technology. An individual compute node consists of a 4th Generation Intel® Xeon® 2 × 32-core-E5-6448Y 32C 2.1 GHz processor, totalling 64 cores per node. Each of the $Re_{\tau} = 350$ cases required a total of 3392 CPU hours to simulate $10T_{\epsilon}$ using one full node. The small-size domain with $Re_{\tau} = 500$ required a total of 4200 CPU-hours to simulate $10T_{\epsilon}$ using one full node, while the medium-size domain with Re_{τ} = 500 cases required a total of 13100 CPU-hours to simulate $10T_{\epsilon}$ using two full nodes, and the large-size domain with $Re_{\tau} = 500$ required a total of 35900 CPU-hours to simulate $10T_{\epsilon}$ using four full nodes. All simulations detailed in Table 2 were run for a total of $10T_{\epsilon}$, and the temporal evolution of the statistics was compared by averaging the statistics for the last $5T_{\epsilon}$, unless otherwise specified.

Table 2: Various simulations considered in this paper. Case naming convention first denotes the friction Reynolds number (e.g., Re500 for a friction Reynolds number of 500.0) followed by the type of initial condition used encoded by the first three letters (e.g., Syn for Synthetic Profile) followed by domain size denomination where S, M, and L are small, medium, and large domain sizes, respectively, and finally the integral length scale used in the iSTFG (e.g., I20 corresponds to $I_l^+ = 20$). All cases except the large domain cases ending with the letter L have a spatial resolution $\Delta x_1^+ = 4.18$, $\Delta x_2^+ = 3.0$, $\Delta x_{3,min}^+ = 0.4$, while the large domain cases have a spatial resolution $\Delta x_1^+ = 6.14$, $\Delta x_2^+ = 4,18$, $\Delta x_{3,min}^+ = 0.4$, respectively. For cases with $Re_{\tau} = 350$, $\Delta x_{3,max}^+ = 5.4$, while for cases with $Re_{\tau} = 500$, $\Delta x_{3,max}^+ = 3.7$. This resolution is sufficient for channel flows to resolve the requisite flow features of interest [40,41]. For both the friction Reynolds numbers, the synthetic profiles are obtained from the dataset by Moser et al. [20].

Case name	Re_{τ}	Initial conditions	Grid	Domain Size	I_l^+	CPUs
Re350LinM	350	Linear Profile	1048x764x128	4pH x 2pH x H	-	64
Re350LogM	350	Log. Profile	1048x764x128	4pH x 2pH x H	-	64
Re350SynM	350	Synthetic Profile	1048x764x128	4pH x 2pH x H	100	64
Re500LinM	500	Linear Profile	1500x1048x256	4pH x 2pH x H	-	128
Re500LogM	500	Log. Profile	1500x1048x256	4pH x 2pH x H	-	128
Re500SynM	500	Synthetic Profile	1500x1048x256	4pH x 2pH x H	100	128
Re500LinS	500	Linear Profile	750x524x256	2pH x pH x H	-	64
Re500LogS	500	Log. Profile	750x524x256	2pH x pH x H	-	64
Re500SynS	500	Synthetic Profile	750x524x256	2pH x pH x H	100	64
Re500LinL	500	Linear Profile	2048x1500x256	8pH x 4pH x H	-	256
Re500LogL	500	Log. Profile	2048x1500x256	8pH x 4pH x H	-	256
Re500SynL	500	Synthetic Profile	2048x1500x256	8pH x 4pH x H	100	256
Re500SynMI20	500	Synthetic Profile	1500x1048x256	4pH x 2pH x H	20	128
Re500SynMI40	500	Synthetic Profile	1500x1048x256	4pH x 2pH x H	40	128

Re500SynMI60	500	Synthetic Profile	1500x1048x256	4pH x 2pH x H	60	128
Re500SynMI80	500	Synthetic Profile	1500x1048x256	4pH x 2pH x H	80	128

4.1 Medium domain size convergence behaviour

4.1.1 Convergence history for the mean velocity and variances

First, we present the convergence of the shear stress as a function of time, as it represents the global balance between the imposed pressure gradient and shear stress at the wall. Thus, the shear/friction velocity (u_τ) can be deduced from the driving pressure gradient and the specified channel height.

$$u^2{}_\tau = \Pi_c H \qquad (15)$$

Here, it is assumed that the shear-stress has units of m²⁄s², which is equivalent to setting $\tau = \tau/\rho_0$, where ρ_0 is the density of the fluid.

Figure 12 compares the time evolution of the normalised shear stress for the three initial conditions and two Re_τ considered in this study. For cases with the log profile, the transition occurs identically for both values of Re_τ such that initially, the shear stress is approximately half of the target value, followed by a sudden transition to elevated shear stress due to the downward convecting pair of vortices that trigger the flow to a turbulent state. Despite scaling the vortex pair and the initial condition by the bulk velocity associated with the target flow Reynolds number, the shear stress still experiences an overshoot that requires more than $5T_\epsilon$ to reach close to the ±5% of the target value. Comparing the behaviour of the vortex pair with the linear profile results in an opposite trend, where the shear stress has a large magnitude that effectively triggers the flow into a turbulent state earlier than the log profile. However, once the transition occurs, the shear stress is lower than observed for the log profile cases. This difference can mainly be attributed to the fact that in the linear profile, once the flow transitions to turbulence, the largest velocities close to the wall are suppressed due to momentum mixing in the vertical direction. This results in smaller shear stress. In the log profile, the region close to the bottom wall exhibits a relatively higher velocity, thus resulting in an overall larger shear stress. For the synthetically generated initial conditions, a similar lower shear stress is observed within the first eddy turnover, with a slight overshoot above the target value for case Re350SynM, which eventually asymptotes to the target value. Since the synthetically generated turbulence retains the scaled components of all the fluctuating quantities, the flow is much closer to the target state and transitions to ±5% of the target value in $2T_\epsilon$.

For case Re500SynM, it is observed that the shear stress is consistently smaller than the target value in the transient phase, while being within ±5% of the target value. However, despite this minor difference, the overall trend remains identical to case Re350SynM. Using a linear fit to the data after an initial transient of $5T_\epsilon$, the log profile is expected to converge at around $21.5T_\epsilon$ and $19.0T_\epsilon$. In comparison, the linear profile is expected to converge at $19.0T_\epsilon$ and $15.0T_\epsilon$ for $Re_\tau = 350$ and $Re_\tau = 500$, respectively. The same linear fit suggests that for the log profile to first enter the ±5% range, it would take a total of $\sim 11T_\epsilon$, while for the linear profile, it would take $\sim 12T_\epsilon$. These linear fits are crude estimates based on the overall trend after $5T_\epsilon$. In addition to the three initial conditions discussed here, we also tested another standard technique to spin up the simulation, where a precursor simulation is run at a smaller Reynolds number or a coarse grid; typically ${Re_\tau}^p = {Re_\tau}^t/2$, where $Re^p{}_\tau$ is the precursor friction Reynolds number and $Re^t{}_\tau$ is the target friction Reynolds number. Typically, the precursor simulation is run for $3T_\epsilon$, and then the velocity and pressure fields are mapped to the target Reynolds number simulation grid. The precursor method requires large computational resources to obtain the initial conditions for the target simulation and a longer time to converge to the statistically stationary flow state. In our comparison, we only considered an initial spin-up of $3T_\epsilon$ for the precursor method. It is essential to note that this may not be sufficiently long to obtain converged statistics before mapping the solution to the target simulation. However, this choice was made with the aim of allowing for a fair comparison between the iSTFG and the precursor method to obtain the initial conditions. Since the precursor method solves the governing equations while iSTFG merely samples a signal, the primary differentiator between the two is the algorithmic complexity. Thus, the computational cost for the precursor simulation is relatively higher and was limited to $3T_\epsilon$. Since the precursor method was not the focus of this study, a visual comparison of the various initial conditions and the corresponding velocity profiles can be visualized in the supplementary movies included with the manuscript.

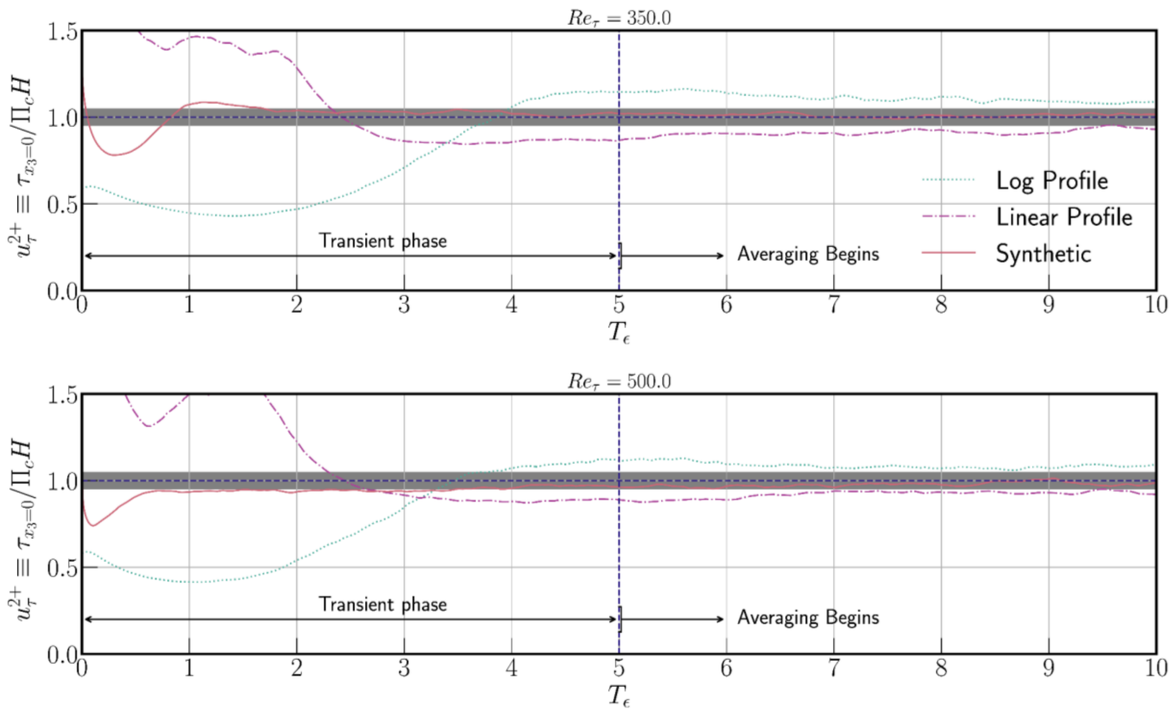

Figure 12: Comparison of the convergence of u_τ^2 for the various initial conditions and Re_τ discussed in this study. The grey region marks the ±5% around the target value.

Figure 13 compares the mean and rms velocity profiles for the three initial conditions detailed in the previous section. All profiles discussed below are averaged over the last $5T_\epsilon$ as detailed in Figure 12 with the right-pointing arrow. Since the mean velocity undergoes significant changes as a function of time during the transient phase, the time-averaged velocity changes as a function of the averaging window. Consequently, for a consistent comparison, the velocity is decomposed as

$$U_i(x_i,t) = < U_i > (x_3) + u_i{}'(x_i,t) \quad (16)$$

Where, $\langle U_i \rangle$ is the plane- and time-average using the last $5T_\epsilon$. While the convergence of the shear stress provides a first indication of the stationary state of the flow, the velocity profiles and their variances also need to converge to carry out meaningful averages. As seen in the plane- and time-averaged velocity ($\langle U_1 \rangle$), excellent agreement between the expected analytical log-law (black dashed line) and case Re350SynM is observed, as expected, while case Re350LogM shows an elevated velocity profile, and case Re350LinM shows a slightly lower velocity profile. Case Re350SynM exhibits a slightly higher velocity magnitude in the inertial range, which can be primarily attributed to the relatively low Reynolds number and the averaging time used to compute the statistics compared to standard datasets. Regardless, swift, statistically stationary conditions are expected based on the convergence history of the shear stress, as detailed in Figure 12. The synthetic profile compares well with the data from Moser et al. [20], further validating the convergence history observed in the shear stress. As the case Re350LogM shows, the entire averaging period exhibits elevated shear stress, resulting in a mean velocity profile that suffers from an excess of total kinetic energy, which requires a longer time to dissipate. For case Re350LinM, the opposite trend is observed and thus exhibits a relatively lower velocity magnitude. The rms velocity profiles and the Reynolds stress exhibit a similarly consistent trend, as observed in the mean velocity profile for all the cases with $Re_\tau = 350$.

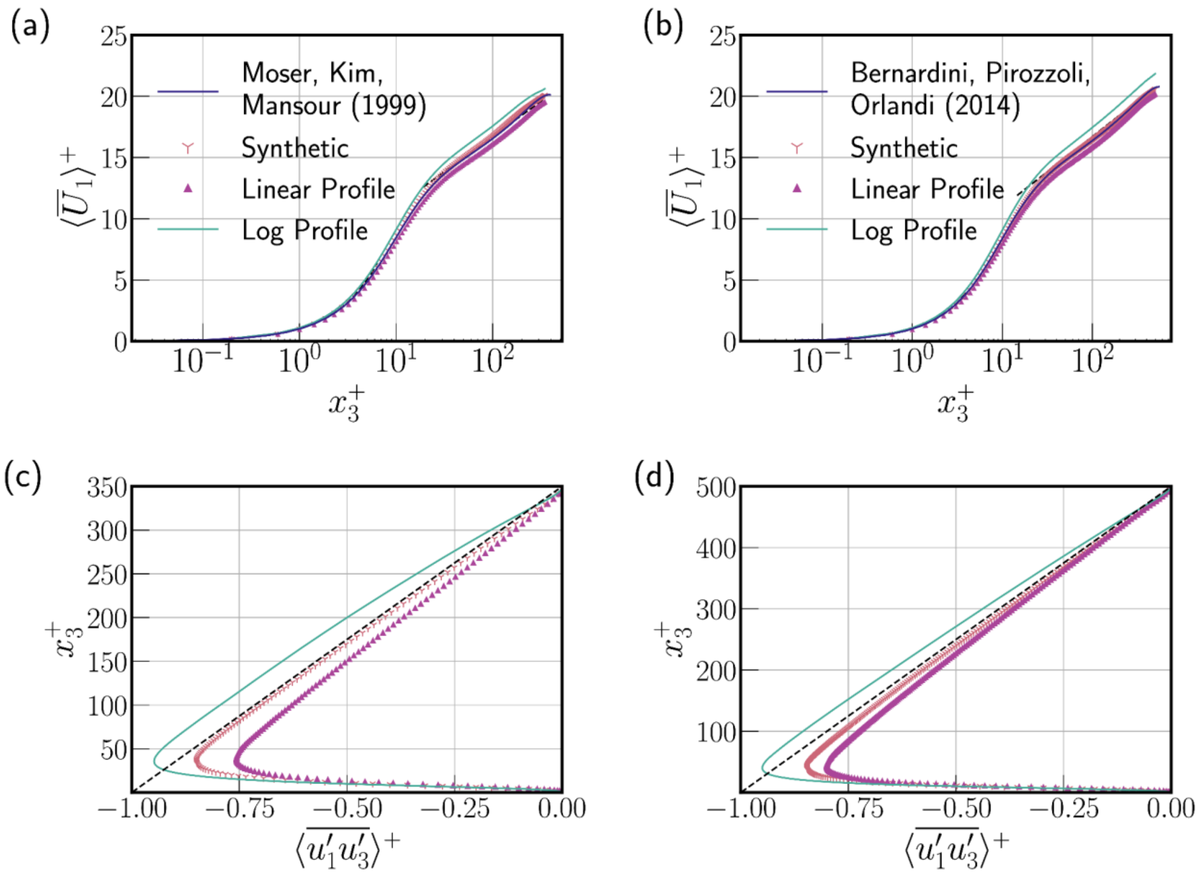

Figure 13: (a) Comparison of plane- and time-averaged velocity profiles with $Re_\tau = 500$ for the three initial conditions against the reference dataset by Moser et al. [20]. (b) Comparison of plane- and time-averaged velocity profiles with $Re_\tau = 500$ for the three initial conditions against the reference dataset by Bernardini et al. [40]. The dashed line in panels (a) and (b) corresponds to the log-law detailed in Eq. (5). Panels (c) and (d) compare the Reynolds stress profiles for the three initial conditions considered in this paper, where panel (c) corresponds to $Re_\tau = 350$ and panel (d) corresponds to $Re_\tau = 500$.

For cases with $Re_\tau = 500$, a similar trend is observed when the three initial conditions are compared against each other. Specifically, case Re500SynM shows an excellent agreement with the dataset from Bernardini et al. [40] for the mean velocity profile, while cases Re500LogM and Re500LinM exhibit a larger and smaller velocity magnitude, respectively, compared to the expected profiles. This trend between the three initial conditions is consistent across the rms velocity profiles and the Reynolds stress profiles, suggesting that the synthetic initial condition is expected to converge identically for higher friction Reynolds numbers. The rms velocity profiles and Reynolds stress are not compared against the dataset as the top wall boundary conditions are not identical, and minor differences can be observed when compared to a no-slip and free-slip channel. However, this does not affect the overall convergence trend presented in this work. Although not explicitly shown here, the absolute error between the synthetic profiles and the benchmark datasets across the entire velocity profile is less than 3%. These observations collectively propose that the mean velocity profiles and the variances converge relatively quickly for the synthetic initial condition compared to the analytical profiles discussed in this paper. In the following section, the convergence will be assessed for higher-order statistics, such as spectral energy and turbulent kinetic energy budgets.

4.1.2 Time evolution of the energy spectrum and turbulent kinetic energy budget

The time evolution of the energy spectrum complements the observations made in the previous subsection. Specifically, in this case, we look at the plane-averaged spectrum for the velocity magnitude as detailed in Figure 14. A consistent trend between the three initial conditions is observed for both values of Re_τ, where the linear profiles exhibit relatively smaller total energy at a given wave number compared to the synthetic and logarithmic profiles, however, the difference between the three cases is not large. The time evolution of the spanwise- and plane-averaged streamwise spectral energy for cases Re350LogM and Re500LogM exhibits consistent behaviour, where the total energy during the first $5T_\epsilon$ is relatively lower, increasing to a consistent value. The time evolution of the spectral energy for cases Re350LinM and Re500LinM also shows a consistent transition to turbulence. A key difference between the linear and log profiles is that the excursion around the mean value for the linear profile is relatively smaller compared to the one observed for the log profile. As for cases Re350SynM and Re500SynM, they do not appear to deviate significantly from their initial state, and most of the changes are observed at high wavenumbers. In contrast, the small wavenumbers corresponding to the large-scale turbulent features of the flow are relatively converged. Since the synthetic initial condition preserves the form of the Reynolds stress tensor using the exponential kernel in space, the distribution of spectral energy is preserved at the respective wavenumbers, unlike the analytical profiles where uncorrelated white noise is added on top of a mean profile. This is clearly seen in the spectrum shape at $T_\epsilon = 0$ compared to the averaged spectrum marked with black "+" symbols.

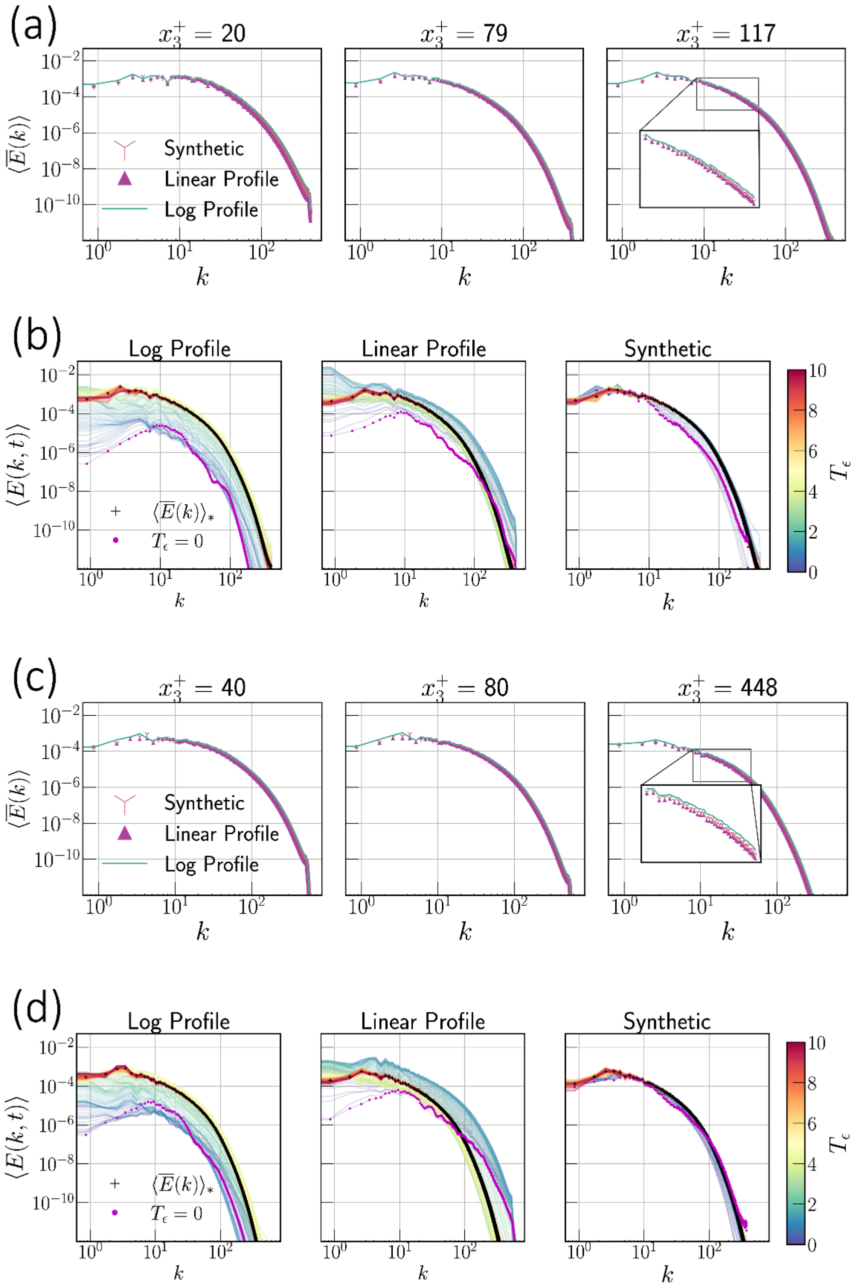

Figure 14: Comparison of spanwise- and time-averaged energy spectra for Re_τ= 350 (panel a) and $Re_\tau = 500$ (panel c). Panels (b) and (d) compare the time evolution of the spectra at non-dimensional vertical coordinates 204 and 445 for the three initial conditions discussed in this paper for $Re_\tau = 350$ and $Re_\tau = 500$, respectively. The $\langle \cdot \rangle *$ operator signifies a plane-averaged quantity after the first $5T_\epsilon$ as marked in Figure 12. The x-axis on all panels marks the wavenumber (k) while the y-axis denotes the energy as a function of the wavenumber ($E(k)$).

The turbulent kinetic energy (TKE) budget for a channel flow with a streamwise driving pressure gradient and homogeneous streamwise and spanwise directions is given by,

$$\partial_t K = P_k - \epsilon_k + \nu_k - T_k - \Pi_k \qquad (17)$$

Where terms from the left are the time rate change of TKE, production of TKE by mean shear, TKE dissipation rate, viscous diffusion of TKE, turbulent transport of TKE, and pressure-diffusion of TKE, respectively. The last three terms are divergence terms and only transport the TKE within the domain without changing the net flux of TKE within the system. Figure 15 compares the various terms in the TKE budget that are averaged in the homogeneous directions and over $5T_\epsilon$ as indicated by the right arrow in Figure 12. A consistent trend observed for the mean velocity and its variances is seen for the TKE budget terms; specifically, the TKE production, which is governed by the product of the Reynolds stress and the mean velocity gradient, has a larger magnitude for the log profile cases and a smaller magnitude for the linear cases.

The synthetic initial condition is observed to follow the baseline reference data quite accurately, with case Re350SynM exhibiting a relatively small overshoot, mainly due to the larger mean velocity estimates compared to the reference dataset of Moser et al. [20]. The TKE dissipation rate aligns closely with the baseline dataset for both Reynolds numbers, particularly for the synthetic initial conditions. At the same time, the other two cases seem to differ quite substantially. The divergence terms in the TKE budget also compare well with the synthetic case despite the relatively short averaging and simulation window, suggesting that all the parameters of interest in the channel flow are adequately converged. Having demonstrated the relatively quick convergence to statistically stationary flow state using the iSTFG and medium-sized computational domain, in the following subsection, we explore the sensitivity of iSTFG as a function of the computational domain size and the integral length scale (which acts as a model parameter) on the convergence using the $Re_\tau = 500$ in the following subsections.

.

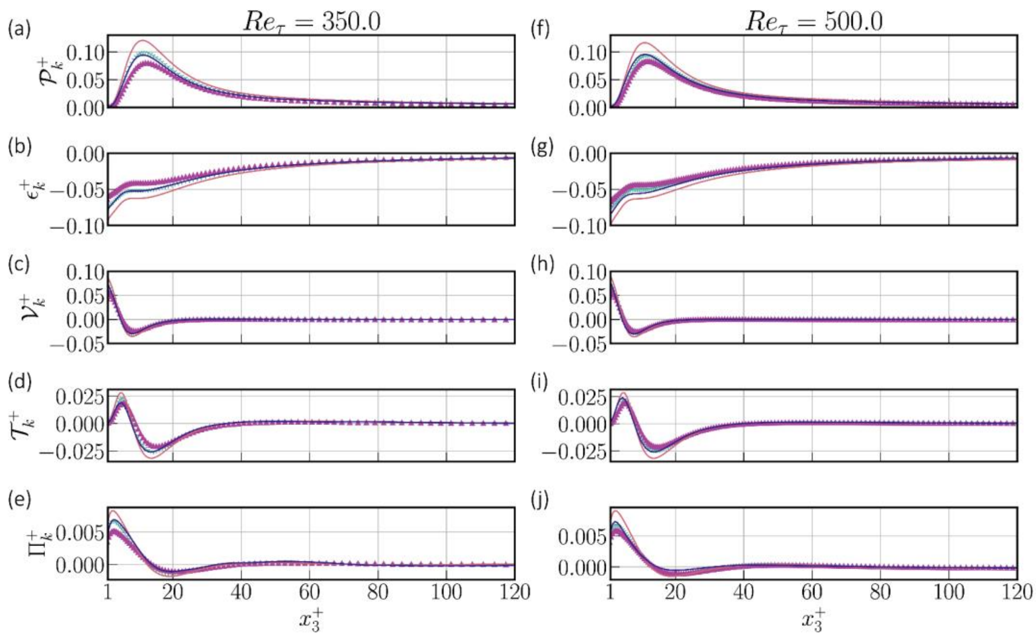

Figure 15: Comparison of the TKE budget terms where all the terms are non-dimensionalised using $u_\tau^4/(\kappa\nu)$. Panels (a)–(e) correspond to $Re_\tau = 500$. Line style identical to Figure 13.

4.2 Effect of domain size and integral length scale on convergence

To better understand the effect of domain size, we consider three different cases as detailed in Table 3 for cases with $Re_\tau = 500$. The time evolution of the plane-averaged shear stress is presented in Figure 16, where for the synthetic profile, the convergence to the target value is independent of the domain size. For cases with a linear profile as their initial condition, a delay is observed in the time at which the transition occurs, and this delay increases with increasing domain size. A similar trend is observed for the cases with the log profile as their initial condition. As these two initial conditions rely on the vortex pair to trigger the turbulence in the channel, with increasing domain size, the flow requires relatively more time to reach the target state. Figure 17 shows this sensitivity to the domain size, where all the synthetic profile cases (panels c, f, and i) are observed to have transitioned to a turbulent state. However, when comparing the linear profile cases, i.e., Figs. 17(a), (d), and (g), further elucidate the differences observed in the time evolution for this case as seen in Figure 16. A similarly strong effect of the large domain is observed for the log profile cases (panels b, e, and h), where case Re500LogL exhibits a typical turbulent spot [41,42] due to the downward descending pair of vortices that eventually triggers the flow into the turbulent state.

This strong dependence on the domain size for the log profile cases is expected, as only one pair of vortices is introduced in the domain as the initial condition. One could consider adding multiple pairs of such downward-descending vortices; however, as observed in the time evolution of the plane-averaged shear stress in Figure 16, the combination of the log profile and the vortices lead to elevated levels of TKE within the flow domain, which can take a long time to dissipate [11]. The synthetic profile cases, on the other hand, are not affected by variations in the domain size, as the flow field is generated throughout the domain without dependence on local transition or trigger mechanisms; this makes the iSTFG more appealing when a large domain and Re_τ are considered. This discussion illustrated that the synthetic method using the iSTFG is domain-size agnostic when comparing the convergence to a statistically stationary flow state.

Table 3: Computational cost for the various cases used to investigate the impact of I_l^+ on the convergence of flow statistics. The iSTFG was compiled using the OpenMPI mpif90 compiler [46] on a single-socket 13th Generation Intel® Core™ i9-13900K CPU with 64 GiB of Random-Access-Memory (RAM) using 8 CPUs.

Case Name	Wall clock (s)	CPU-hours
Re500SynMI20	71	0.16
Re500SynMI40	117	0.26
Re500SynMI60	210	0.48
Re500SynMI80	295	0.66
Re500SynMI100	414	0.92

Since the synthetic profile is independent of the domain size used to obtain the time evolution, the effect of the integral length scale used in the iSTFG as the only model parameter was also explored to quantify its sensitivity. For turbulent channel flows, a wide range of detailed studies has been conducted to better understand the transition to a turbulent flow state [18, 44, 45], elucidating the various mechanisms that lead to this transition.

.

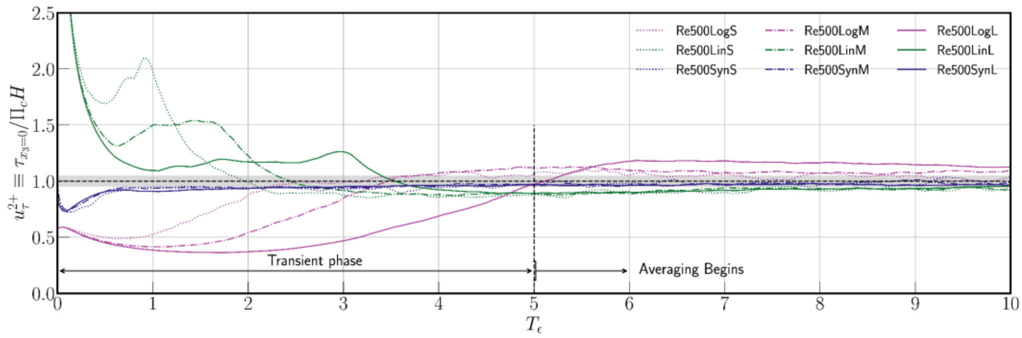

Figure 16: Comparison of the convergence history as a function of the initial condition and the domain size used to simulate the channel flow.

For a statistically stationary flow state, a fundamental length scale of interest is the integral length scale I_l^+, which for channel flows has been suggested to be $I_l^+ \sim 100$ [18,34]; corresponding to the streamwise streaks responsible for near-wall turbulence intermittency [18]. It has also been demonstrated that this is the relevant scale of interest when considering the non-linear regeneration cycle that sustains turbulent motion within the buffer layer of wall-bounded flows, thereby proving to be a useful scale for testing the sensitivity of iSTFG.

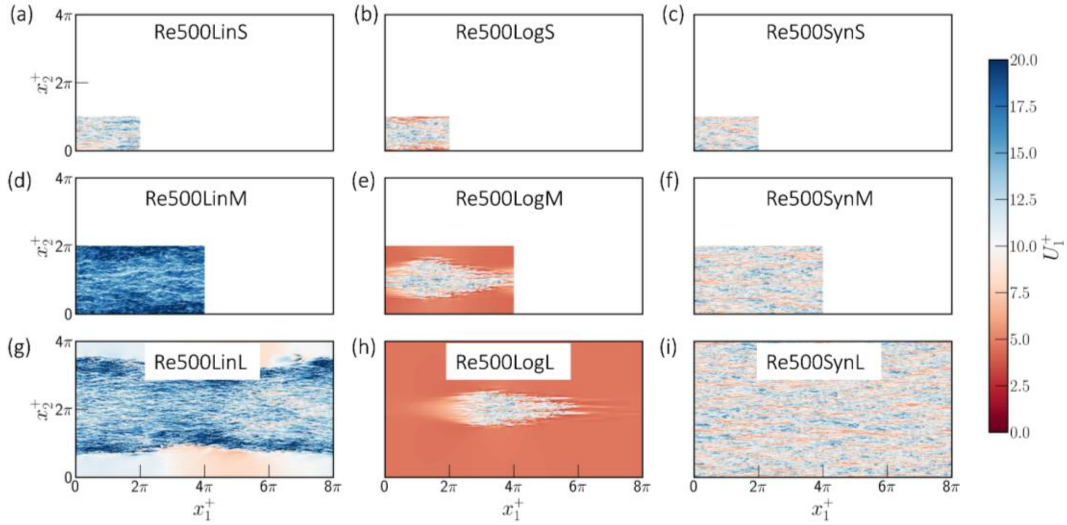

Figure 17: Normalised streamwise velocity heatmap comparison at x_3 = $H/8$ above the bottom wall at $T_\epsilon = 2$ for the cases marked at the top centre on each panel of the figure.

To this end, we chose the medium-sized computational domain to investigate the effect of varying values of I_l^+ as detailed in Table 2. Since the iSTFG's computational cost scales with the width of the isotropic stencil used in generating the field, larger values of I_l^+ for all other identical parameters result in a higher computational cost. For cases Re500SynMLI20, Re500SynMLI40, Re500SynMLI60, Re500SynMLI80, and Re500SynMLI100, the computational cost to generate the turbulent initial conditions is listed in Table 3. It can be readily concluded from the data presented in Table 3 that smaller values of I_l^+ correspond to relatively lower computational costs. Thus, smaller values of I_l^+ are preferred. Comparing the time evolution of the plane-averaged shear stress shown in Figure 18 suggests that most differences observed for varying values of I_l^+ are during the early part of the transient phase. For case Re500MI20, the time evolution is like that of case Re500LogM primarily because the I_l^+ sets the correlation length scale. Consequently, with decreasing values of I_l^+, the flow decorrelates relatively quickly and tends to the log profile superposed with white noise as seen in Figure 19. The use of an isotropic I_l^+ value for all the velocity components can be clearly seen in Figure 19. During the initial part of the transient phase, there is a weak dependence observed as a function of the I_l^+. Smaller values of I_l^+ seem to transition relatively later when compared to the larger values of I_l^+; however, this trend asymptotes to the target value beyond $3T_\epsilon$. These observations suggest that the iSTFG initial condition does not impact the overall time evolution of the shear stress development and convergence to the statistically stationary flow state.

As discussed in the above subsection, the time evolution of the plane-averaged shear stress is observed to be domain size and I_l^+ agnostic for the initial conditions generated using iSTFG; thus, we will briefly discuss the absolute error for the plane-averaged streamwise velocity for the various I_l^+considered in this paper. The absolute error (in %) is calculated using.

$$\epsilon_U = abs\left(U_{ref} - \langle U_1\rangle\right)\left(100U_{ref}^{-1}\right) \qquad (18)$$

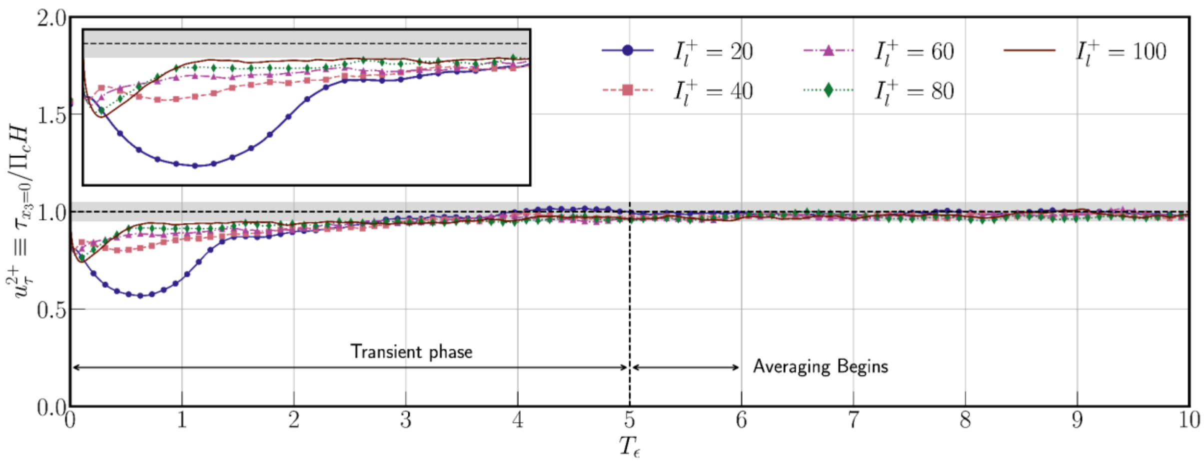

Figure 18: Time evolution of the plane-averaged shear stress as a function of varying values of $I_l^+$for the medium-sized computational domain. The inset shows a zoomed-in view of the first $T_\epsilon = 2.5$ for the various cases. Data markers for the simulations are displayed once every 50 data points.

Where ϵ_U is the per cent error, U_{ref} is the reference velocity profile, and $\langle U_1 \rangle$ is the plane-averaged velocity profile at a given time instance. Figure 20 compares the absolute error for each plane-averaged velocity profile at ten different eddy turnovers as a function of the distance from the bottom wall. Over the first three eddy turns, the error for case Re500SynMI100 exceeds 5% near the bottom wall, while above $x_3^+ > 100.0$, the error decreases rapidly. As for the other cases compared in Figure 20, the error decreases significantly over the first three eddy turns in the near-wall region and stays consistently lower than that of case Re500SynMI100. Far from the wall, the error increases for case Re500SynMI20 but falls below 1% at $5T_\epsilon$. Smaller values of $I_l^+$are observed to exhibit relatively lower absolute error during the transition phase; however, this trend is not as consistent or strong as the absolute error for case Re500SynMI80, which is relatively larger when compared to Re500SynMI100 at $5T_\epsilon$ Overall, all the cases observed exhibit good agreement with the maximum absolute error at $5T_\epsilon$, which is 3% throughout the vertical extent.

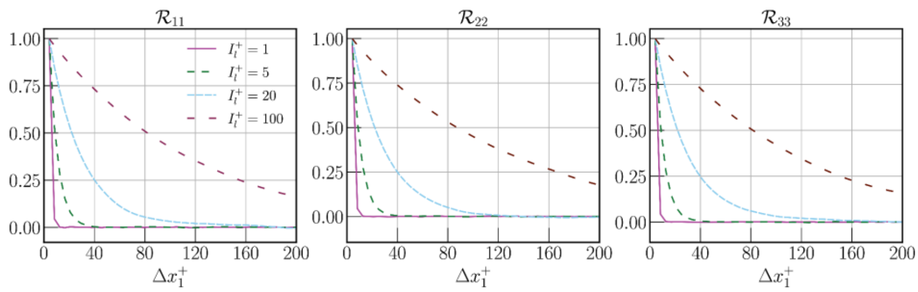

Figure 19: Auto-correlation of the streamwise (R_{11}), spanwise (R_{22}), and vertical (R_{33}) velocity components for four distinct values of I_l^+ as a function of the streamwise spacing. The autocorrelation data presented in this figure were generated for case Re500SynM, where $I_l^+ = 1$ and $I_l^+ = 5$ are only presented for qualitative comparison and were not simulated in this work.

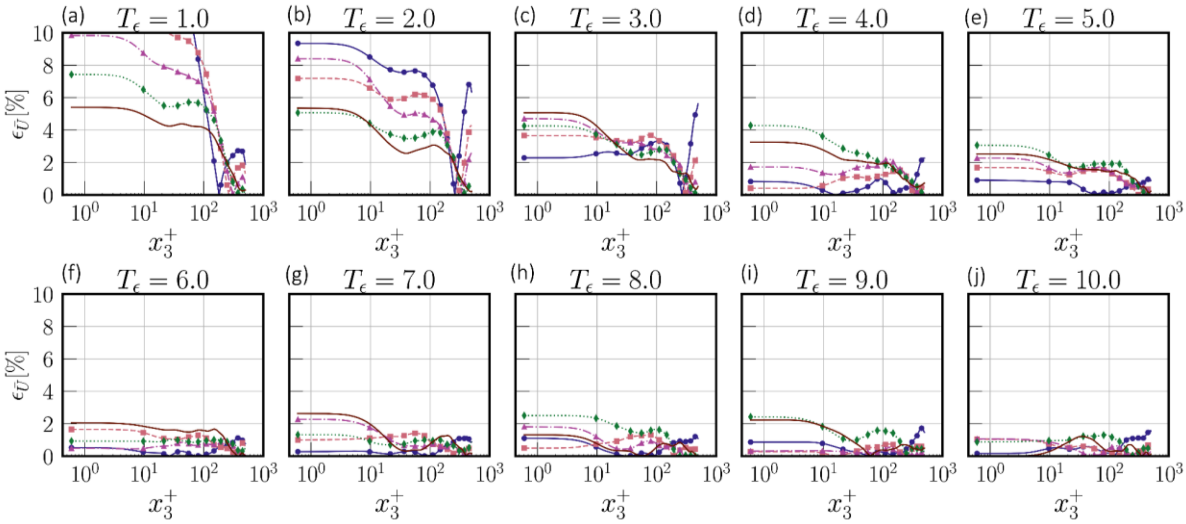

Figure 20: Absolute error (in %) as a function of the eddy turnover time for the medium-size domain with varying I_l^+. The reference velocity profile is obtained by running case Re500SynM for $20T_\epsilon$ and averaging over the last $10T_\epsilon$. Data markers for the simulations are displayed every 20 data points, with the line colour and style identical to those in Figure 18.

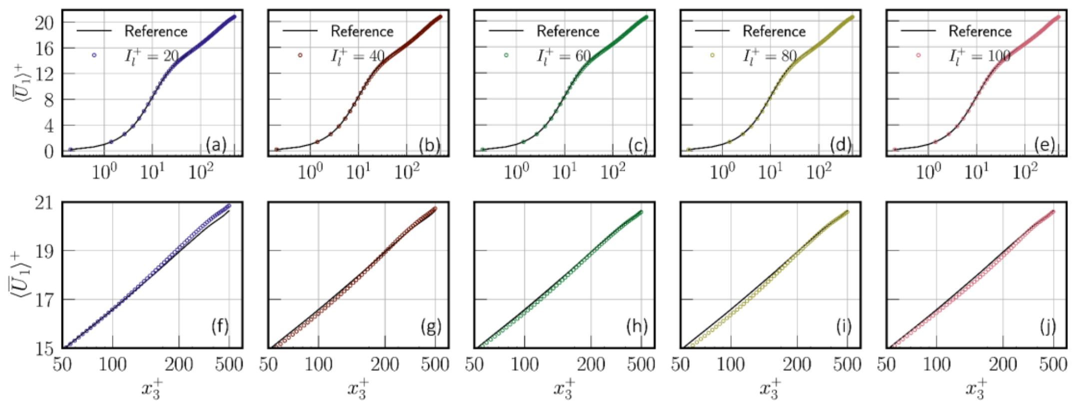

Figure 21: Plane- and time-averaged velocity profile comparison for various values of the model parameter I_l^+simulated in this paper. The top row panels (a-e) show the entire velocity profile, while the bottom row panels (f-j) show the log-law region of the velocity profile. The velocity profiles are time-averaged after the initial five eddy turns. Data markers for the simulations are displayed for every third data point.

Figure 21 compares the plane- and time-averaged velocity profiles against the reference velocity profile. Consistent with the shear stress convergence as a function of time, increasing values of I_l^+ are observed to have relatively more significant errors when compared against the reference data throughout. This is largely supported by the previous discussion when the absolute error is compared in Figure 20. Additionally, cases with relatively small values of I_l^+ are observed to converge quickly in the lower portion of the log-law region. In contrast, closer to the top boundary, cases with relatively large values of I_l^+ are in better agreement with the reference data, as evidenced by Figs. 20 and 21. It is pertinent to recognise that the results presented in Figure 21 are subject to improvement with increasing averaging time used to obtain the plane- and time-averaged results. These observations, combined with the computational cost associated with iSTFG as a function of I_l^+ (see Table 3), suggest that of the five values for I_l^+ discussed in this paper, $I_l^+ = 20$ is the computationally efficient choice when all other parameters are identical. These results provide further support for the utility of the iSTFG in achieving fast and accurate stationary flow conditions without incurring a large computational cost.

4.3 Impact of friction Reynolds number mismatch

In the previous sections, a detailed discussion was presented to motivate the utility of the iSTFG in obtaining converged statistics using reference data from precomputed channel flow datasets [29, 40, 47, among others]. However, it is easy to imagine a situation where such reference data is either unavailable or contains a friction Reynolds number mismatch. To make iSTFG functional in scenarios like this, we conducted a sensitivity test for iSTFG to achieve a targe $Re_\tau = 500$, subject to an input data mismatch as detailed in Table 3. Specifically, we compared the convergence to an a priori known target, $Re_\tau^t = 500$, with initial conditions having 10% and 20% deviations. For all cases detailed in Table 3, the model parameter $I_l^+ = 20$ corresponds to the reference friction Reynolds number (Re_τ^r.). The CPU hours for the various cases differ as the underlying computational grid is designed to resolve the requisite scales of motion for Re_τ^t. Thus, for a relatively lower value of Re_τ^r, the stencil width for a fixed model parameter of I_l^+ spans over relatively more grid points, resulting in a slightly higher computational cost and vice versa. For case Re500SynMI20-450, no reference dataset is available from conventional channel flow simulations. As a result, the mean velocity and Reynolds stress tensor profiles are obtained by rescaling the friction Reynolds number using data from Moser et al. [20]. For the last two cases in Table 4, the time and plane-averaged velocity can introduce a large mismatch due to the Re_τ^r. and Re_τ^t. difference. Since the logarithmic profile is assumed to be universal [47], the mean velocity input profile for these three cases is prescribed as per Eq. (5) except for the white noise terms.

Figure 22 compares the effect of Re_τ mismatch on the time evolution of the plane-averaged shear stress for the various cases detailed in Table 3. Since cases Re500SynMI20-395, Re500SynMI20-450, Re500SynMI20550, and Re500SynMI20-590 are not scaled down to the target simulation friction Reynolds number, they clearly exhibit a sensitivity to initial conditions. Specifically, cases initialised with a relatively lower friction Reynolds number ($Re_\tau^r < Re_\tau^t$) exhibit a slower convergence to the target value when compared to the cases initialised with a relatively higher friction Reynolds number ($Re_\tau^r > Re_\tau^t$). This observation can be generalised when the time evolution of the vertically integrated bulk velocity is considered, as shown in Figure 23(a). Cases with a relatively larger Re_τ mismatch take longer to reach the target value, with a systematic trend observed across varying Re_τ values. This observation can be further understood through the vertically integrated TKE and Reynolds stress time evolution shown in Figure 23. Cases that are not rescaled and exhibit a mismatch (cases Re500SynMI20-395, Re500SynMI20450, Re500SynMI20-550, and Re500SynMI20-590) show a wide range of variation in the time evolution of both the TKE and the Reynolds stress. For cases Re500SynMI20-590 and Re500SynMI20-395, the differences are small during the transition but grow after the initial $0.5T_{\epsilon}$, as seen in panels (b) and (c) of Figure 23. More importantly, the relative amount of TKE between cases Re500SynMI20-395 and Re500SynMI20-590 is drastically different, leading to a slower convergence towards the target value.

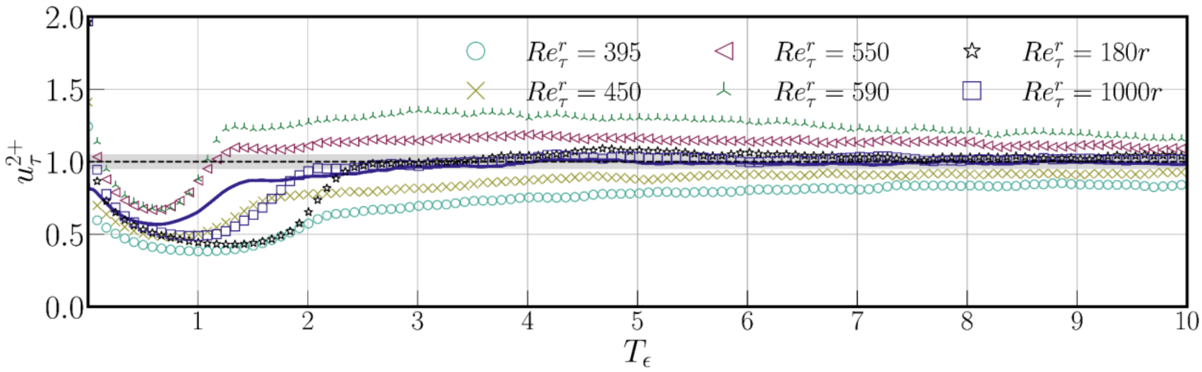

Figure 22: Time evolution of the plane-averaged shear stress as a function of varied Re_τ^r mismatch against the target $Re_\tau^t = 500.0$ for the cases detailed in Table 3. The solid blue line marks the time evolution of the plane-averaged shear stress for case Re500SynMI20, while the black dashed line marks the time-averaged value, with the grey shading denoting the ±5% around this time-averaged value. Data markers for each case are only shown for one in forty points.

As presented in Figs. 22 and 23, the time evolution of the various parameters of interest suggests that iSTFG can reliably accelerate convergence towards a stationary turbulent state when the input data is correctly scaled to the target Re_τ. This aims to solve two challenges associated with turbulent channel flow spinup: (a) transition to a turbulent flow state for all Re_τ[18], (b) relatively fast convergence towards a statistically stationary flow state when compared to conventional methods. Specifically, using iSTFG to generate initial conditions forced only with a constant pressure gradient can accelerate the transition to the statistically stationary flow state. As detailed extensively in the discussion above, iSTFG is not sensitive to the model parameter and mismatch in the available Re_τ reference data if the mean velocity is specified through the classical log-law [48] and the Reynolds stress profiles are scaled to the target Re_τ^t. As detailed in Figs. 22 and 23, when the transition to statistically stationary flow is considered, all cases using the correctly scaled Re_τ^t converge to the target values (±5%) within $5T_\epsilon$ for all parameters of interest at a minimal computational cost.

Table 4: Simulation details for cases used to verify the sensitivity of initial conditions with a Re_τ mismatch between the reference and the target simulations. The case naming convention is identical to the definition detailed in Table 1, with the −⟨num⟩ indicating the Re_τ^r from which the initial conditions are derived. Here, Re_τ^r and Re_τ^t correspond to the reference and the target friction Reynolds number, respectively, while the CPU-hours column denotes the computational cost associated with generating the initial conditions.

Case name	Re_τ^t	Re_τ^r	% mismatch	CPU-hours	Source
Re500SynMI20-395	500	395	~20% lower	0.165	[20]
Re500SynMI20-450	500	450	10% lower	0.153	Rescaled to $Re_\tau = 395$ [20]
Re500SynMI20-550	500	550	10% lower	0.140	[46]
Re500SynMI20-590	500	590	~20% higher	0.132	[20]
Re500SynMI20-180rescaling	500	180	~65% higher	0.140	Rescaled to $Re_\tau = 500$ [20]
Re500SynMI20-1000rescaling	500	1000	100% higher	0.140	Rescaled to $Re_\tau = 500$ [46]

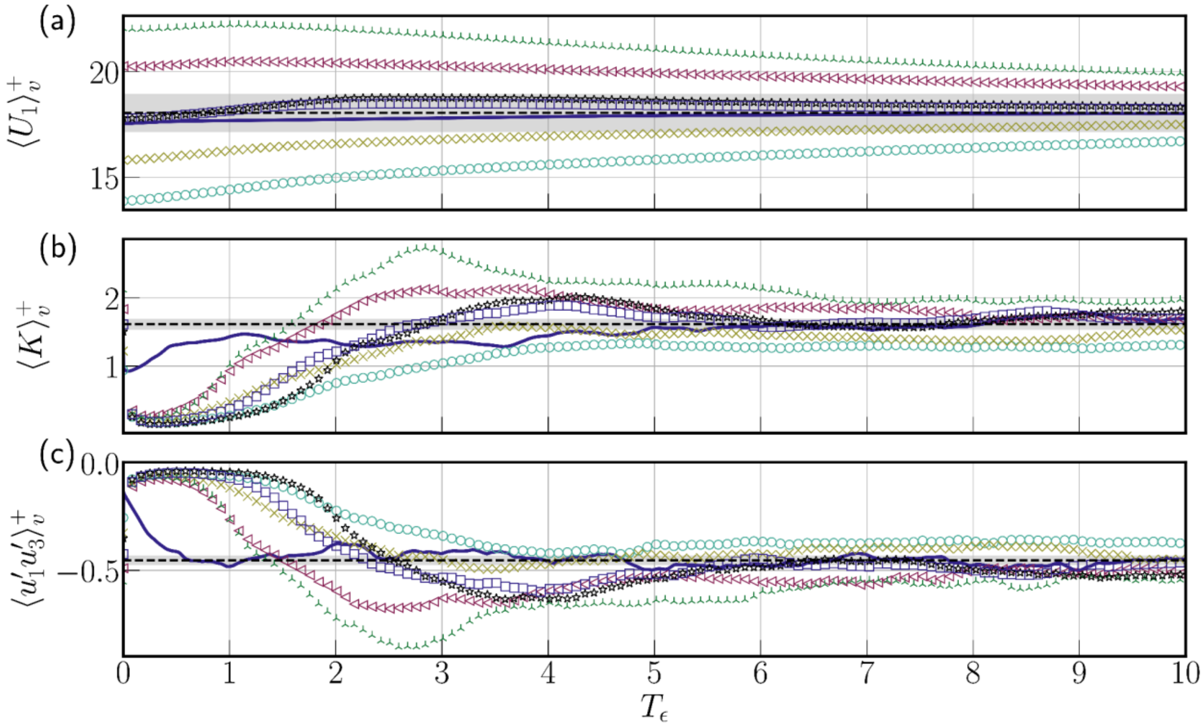

Figure 23: Time evolution of the plane-averaged and vertically integrated profiles (a) streamwise velocity, (b) Turbulent kinetic energy, and (c) Reynolds stress, respectively, as a function of time. The solid blue line on all the panels corresponds to case Re500SynMI20, while the black dashed line marks the time-averaged value, with the grey shading denoting the ±5% around this time-averaged value.

4.4 Limitations of iSTFG

The previous section presented a detailed discussion on the performance of iSTFG to obtain stationary flow conditions. While iSTFG provides a computationally efficient method to synthesise initial conditions, it has limitations that can be further improved, as detailed below.

- All the tests presented in this work were carried out at a relatively modest friction Reynolds number ($Re_\tau = 500$). This was primarily done to mitigate the steep increase in computational cost when conducting the parameter sensitivity test. Although similar results have been obtained for relatively higher $Re_\tau \sim 700$ (not presented in this work), additional numerical experiments are needed to generalise this approach for higher friction Reynolds numbers.
- The input data required for iSTFG must be carefully rescaled to the target friction Reynolds number, as discussed in Section 4.3, as the time required to reach the statistically stationary flow state can be influenced by a Reynolds number mismatch. While this may seem like a severe limitation, appropriate rescaling

yields a similar convergence trend for cases where exact mean velocity profiles and Reynolds stress tensor data are not available.

- One of the strongest limitations of iSTFG in its application to turbulent channel flows is the inherent assumption of streamwise and spanwise homogeneity. In its current form, iSTFG translates all the assumptions underlying classical synthetic eddy methods [33], and a generalised application beyond prototypical channel flows must be explored with caution. Specifically, iSTFG does not provide a generalised solution for problems where the spanwise and streamwise homogeneity is not preserved, thus limiting its application to these equally interesting turbulent flow problems.

5. Description of the multi-fidelity framework and preliminary results

In the previous section, we described the developments required by the pre-processing methods for the higher-fidelity frameworks. Now, we briefly present the multi-fidelity framework that combines these methodologies. We employ Kriging, also known as Gaussian Process Regression (GPR), a powerful surrogate modelling technique that provides both predictions and uncertainty estimates based on spatial or parametric correlations within data [48]. In the context of computational fluid dynamics (CFD), Kriging is particularly effective for constructing surrogate models that approximate high-fidelity simulations while requiring significantly fewer computational resources. By treating the unknown function as a realisation of a Gaussian process defined by a mean and covariance function, Kriging can capture complex nonlinear relationships between input parameters and CFD outputs (e.g., velocity, pressure, or drag coefficients). In our case, the wind direction serves as the input parameter, while the wind-flow response within the region of interest serves as the CFD output. This enables interpolation of sparse high-fidelity simulation data and the generation of confidence intervals for predictions, allowing efficient exploration of high-dimensional parameter spaces. Within a multi-fidelity CFD modelling framework, Kriging acts as a bridge that integrates data from multiple sources of varying accuracy and cost—such as low-fidelity RANS models and high-fidelity DNS or LES simulations—into a unified predictive model. Multi-fidelity Kriging techniques, including co-Kriging, exploit the correlations between low- and high-fidelity datasets to enhance surrogate accuracy while minimising reliance on computationally expensive simulations. This hierarchical fusion enables rapid design optimisation, sensitivity analysis, and uncertainty quantification within complex CFD workflows. Consequently, Kriging-based multi-fidelity frameworks are increasingly being adopted to accelerate model-driven decision-making without compromising predictive performance [48]. In the following subsections, we describe the high-fidelity computational model employed in this study, along with preliminary results corresponding to the predominant wind directions at the TUD campus.

5.1 Description of the high-fidelity computational model

We utilize the uDALES large-eddy simulation framework to simulate the wind around realistic, complex urban areas. The uDALES computational framework is an advanced atmospheric modelling code designed to simulate turbulent flows, heat, and pollutant transport in urban environments[3]. The model solves the filtered incompressible Navier–Stokes equations under the Boussinesq approximation, along with scalar transport and surface energy balance equations, enabling a realistic representation of building–atmosphere interactions. In addition, uDALES includes a full surface-energy balance on urban facets (for heat storage, conduction, radiation, and sensible/latent fluxes) to couple flow–thermal processes. Thus, the model addresses momentum, continuity, scalar transport, plus surface processes. The Boussinesq approximation means that density variations only appear in the buoyancy term. Compressibility effects are neglected, which is appropriate for atmospheric boundary-layer/urban microclimate scales.

The spatial discretisation in uDALES employs a Cartesian grid on an Arakawa C grid arrangement, where velocities are staggered (each velocity component at its corresponding face) and pressure and scalars are located at cell centres. Second-order central finite differences are used for the prognostic fields (velocity, pressure, scalars), except for pollutant concentration advection, where a kappa-advection scheme is available to maintain monotonicity. Time integration is carried out via a third-order Runge-Kutta scheme for the tendency terms. For the pressure-Poisson equation, uDALES employs an FFT-based solver in the horizontal periodic directions and a tridiagonal Gaussian elimination solver in the non-uniform vertical direction, allowing for grid stretching in the vertical direction. Turbulence closure is provided via an implicit LES filter (the grid acts as a filter operator) and a sub-grid scale (SGS) eddy-viscosity model – specifically the Vreman model. IBM is used for buildings/obstacles: the solid-fluid boundary does not need to conform to the grid. At the cells adjacent to the boundary, the fluxes are modified (zero flux of advection/diffusion where appropriate), and wall functions provide shear/heat/moisture exchange. Surface-energy-balance is discretised in each surface facet by dividing the material into layers (along depth) and solving a one-dimensional conduction equation (piecewise quadratic polynomial in depth) coupled to turbulent fluxes on top. The uDALES software is implemented in modern Fortran and parallelised using the MPI library. The domain is decomposed using the 2DECOMP and FFT libraries, which allow scalability across thousands of CPUs.

[3] https://github.com/uDALES/u-dales

5.2 Preliminary results for Delft University of Technology Campus

To simulate the flow around the TU Delft campus, we consider the central campus to be like the one used in the low-fidelity framework for a consistent comparison. Using the best practice guidelines, the computational domain was designed to minimise the effects of the boundary conditions in the region of interest[4]. Using GenSDF, the geometry of the campus is immersed within the computational domain, and a precursor simulation is run to supply the right boundary conditions. This ensures that the inflow plane for the primary simulation can be coupled with the results from a statistically stationary flow profile. Consequently, the atmospheric boundary layer profile with dynamically turbulent inflow conditions can be used without artificially introducing artefacts that would otherwise arise from synthetic inflow methods. One such simulation scenario is illustrated in the figure below, which shows an instantaneous velocity snapshot (for part of the domain) that captures the complex flow response within the urban canopy that can be anticipated. It is worth noting that, although the RANS model yields a time-averaged output, the dynamics of in-canopy flow response are significantly different, even when the inflow boundary has a statistically stationary boundary condition. This is due to the turbulent structures and correlations that play a crucial role in this process. Invariably, this comes at a steep computational cost; however, the current framework will be used sparsely for a few wind directions based on Latin Hypercube sampling of the wind direction probability density function, which can limit the number of wind simulation realisations required for such a study. Consequently, in the final version of the multi-fidelity framework, we will utilise this simulation approach to make risk predictions with a limited computational cost yet sufficient accuracy.

[4] Domain designed according to Franke et al. (2007) considering the upstream and downstream fetch.

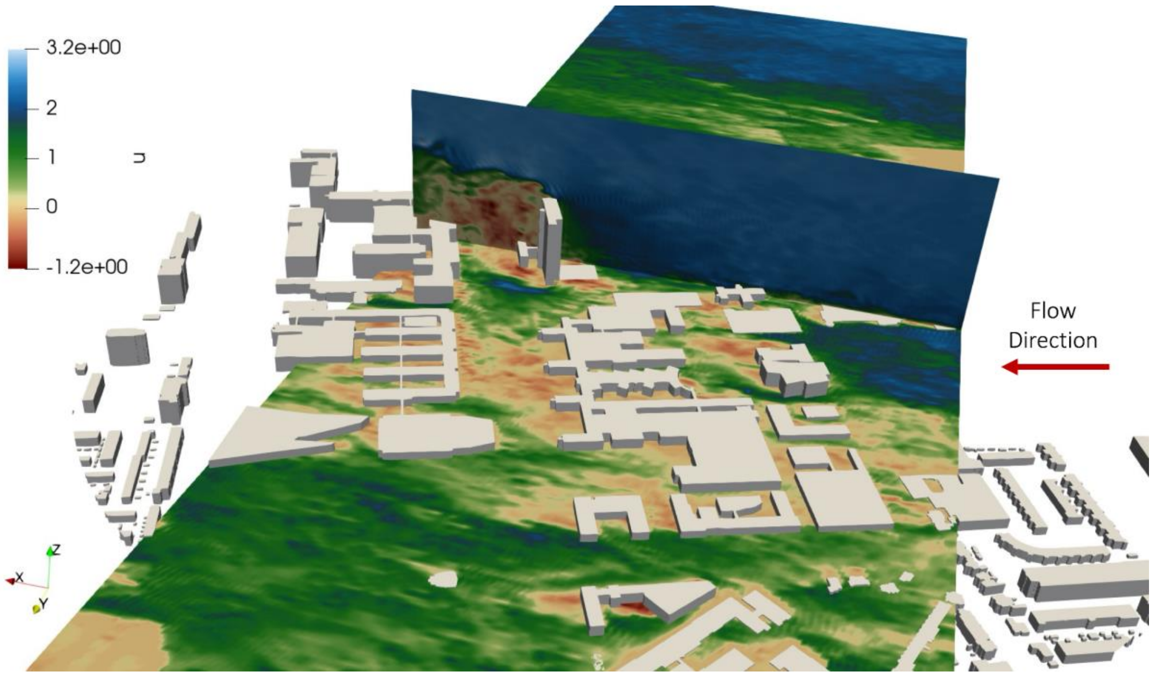

Figure 24: Turbulent velocity snapshot for the TUD central campus for the predominant wind direction.

6 Concluding Remarks

In conclusion, the integration of multi-fidelity modelling strategies represents a significant advancement in the predictive capabilities of urban wind flow simulations, necessitating the development and design of new computational methodologies to smoothly integrate the workflow, thereby allowing for an efficient and practical approach. By effectively coupling high-fidelity, scale-resolving solvers such as uDALES with computationally efficient RANS-based approaches, the developed framework successfully bridges the gap between accuracy and performance. This hybrid methodology enables reliable wind predictions across heterogeneous urban canopies while maintaining tractable computational demands, thus addressing a central challenge in predictive modelling.

The supporting technological developments, namely, the scalable signed-distance-field (SDF) generator and the synthetic turbulence initialisation framework, further strengthen the system's robustness and applicability. The SDF tool offers an efficient method for embedding complex geometries within structured grids, ensuring geometric flexibility without incurring numerical penalties. Meanwhile, the synthetic turbulence generator enhances simulation convergence, effectively reducing the computational overhead associated with achieving statistically steady turbulent flow conditions.

Together, these components substantially improve both the efficiency and fidelity of the computational workflow.

Overall, the resulting Kriging-based multi-fidelity framework establishes a strong foundation for predictive wind modelling. By systematically integrating models of varying resolutions and leveraging data-driven surrogates to manage epistemic and aleatory uncertainty, the approach enables a more comprehensive representation of urban wind. This framework will not only enhance predictive reliability but also offer a scalable pathway for extending such methodologies to broader environmental and climatological applications, supporting sustainable urban design and resilience planning in data-limited scenarios.

.

References

[1] Goc, K.A., Lehmkuhl, O., Park, G.I., Bose, S.T. and Moin, P., 2021. Large eddy simulation of aircraft at affordable cost: a milestone in computational fluid dynamics. *Flow*, *1*, p.E14.

[2] Ciarlatani, M.F., Huang, Z., Philips, D. and Gorlé, C., 2023. Investigation of peak wind loading on a high-rise building in the atmospheric boundary layer using large-eddy simulations. *Journal of Wind Engineering and Industrial Aerodynamics*, *236*, p.105408.

[3] Garcia-Sanchez, C., Philips, D.A. and Gorlé, C., 2014. Quantifying inflow uncertainties for CFD simulations of the flow in downtown Oklahoma City. *Building and environment*, *78*, pp.118-129.

[4] Hochschild, J. and Gorlé, C., 2024. Design and demonstration of a sensing network for full-scale wind pressure measurements on buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, *250*, p.105760.

[5] Jiang, H. and Cheng, L., 2021. Large-eddy simulation of flow past a circular cylinder for Reynolds numbers 400 to 3900. *Physics of Fluids*, *33*(3).

[6] Peskin, C.S., 2002. The immersed boundary method. *Acta numerica*, *11*, pp.479-517.

[7] Yang, J. and Balaras, E., 2006. An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. *Journal of computational Physics*, *215*(1), pp.12-40.

[8] Costa, P., 2018. A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows. *Computers & Mathematics with Applications*, *76*(8), pp.1853-1862.

[9] Roosing, A., Strickson, O.T. and Nikiforakis, N., 2019. Fast distance fields for fluid dynamics mesh generation on graphics hardware. *arXiv preprint arXiv:1903.00353*.

[10] Fortran 2003 Draft International Standard. URL https://wg5-fortran.org/N1601-N1650/N1602.pdf.

[11] Eberly, D., 1999. Distance between point and triangle in 3D. *Magic Software, http://www. magic-software. com/Documentation/pt3tri3. pdf*.

[12] Peters, R., Dukai, B., Vitalis, S., van Liempt, J. and Stoter, J., 2022. Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands. *Photogrammetric Engineering & Remote Sensing*, *88*(3), pp.165-170.

[13] Pađen, I., García-Sánchez, C. and Ledoux, H., 2022. Towards automatic reconstruction of 3D city models tailored for urban flow simulations. *Frontiers in Built Environment*, *8*, p.899332.

[14] Orion capsule, NASA 3D resources. 2025, URL https://nasa3d.arc.nasa.gov/ detail/orion-capsule.

[15] The Stanford 3D scanning repository. 2025, URL http://graphics.stanford.edu/ data/3Dscanrep/.

[16] Li, N. and Laizet, S., 2010, May. 2decomp & fft-a highly scalable 2d decomposition library and fft interface. In *Cray user group 2010 conference* (pp. 1-13).

[17] Patil, A. and García-Sánchez, C., 2025. A comparative hydrodynamic characterization of the flow through regular and stochastically generated synthetic coral reefs over flat topography. *arXiv preprint arXiv:2501.15237*.

[18] The minimal flow unit in near-wall turbulence. *Journal of Fluid Mechanics*, *225*, pp.213-240.

[19] Kasagi, N., Tomita, Y. and Kuroda, A., 1992. Direct numerical simulation of passive scalar field in a turbulent channel flow.

[20] Moser, R.D., Kim, J. and Mansour, N.N., 1999. Direct numerical simulation of turbulent channel flow up to Re= 590. *Phys. fluids*, *11*(4), pp.943-945.

[21] Marusic, I., McKeon, B.J., Monkewitz, P.A., Nagib, H.M., Smits, A.J. and Sreenivasan, K.R., 2010. Wall-bounded turbulent flows at high Reynolds numbers: recent advances and key issues. *Physics of fluids*, *22*(6).

[22] Jiménez, J., 2012. Cascades in wall-bounded turbulence. *Annual Review of Fluid Mechanics*, *44*(1), pp.27-45.

[23] Smits, A.J. and Marusic, I., 2013. Wall-bounded turbulence. *Physics Today*, *66*(9), pp.25-30.

[24] Lozano-Durán, A., Bae, H.J. and Encinar, M.P., 2020. Causality of energy-containing eddies in wall turbulence. *Journal of Fluid Mechanics*, *882*, p.A2.

[25] Jie, Y., Cui, Z., Xu, C. and Zhao, L., 2022. On the existence and formation of multi-scale particle streaks in turbulent channel flows. *Journal of Fluid Mechanics*, *935*, p.A18.

[26] Horwitz, J.A.K., 2024. Estimating the carbon footprint of computational fluid dynamics. *Physics of Fluids*, *36*(4).

[27] Vela-Martín, A., Encinar, M.P., García-Gutiérrez, A. and Jiménez, J., 2021. A low-storage method consistent with second-order statistics for time-resolved databases of turbulent channel flow up to Re_τ= 5300. *Journal of Computational Science*, *56*, p.101476.

[28] Nelson, K.S. and Fringer, O.B., 2017. Reducing spin-up time for simulations of turbulent channel flow. *Physics of Fluids*, *29*(10).

[29] Graham, J., Kanov, K., Yang, X.I.A., Lee, M., Malaya, N., Lalescu, C.C., Burns, R., Eyink, G., Szalay, A., Moser, R.D. and Meneveau, C., 2016. A web services accessible database of turbulent channel flow and its use for testing a new integral wall model for LES. *Journal of Turbulence*, *17*(2), pp.181-215.

[30] Henningson, D.S. and Kim, J., 1991. On turbulent spots in plane Poiseuille flow. *Journal of fluid mechanics*, *228*, pp.183-205.

[31] Kim, J. and Moin, P., 1985. Application of a fractional-step method to incompressible Navier-Stokes equations. *Journal of computational physics*, *59*(2), pp.308-323.

[32] Ferziger, J.H., Perić, M. and Street, R.L., 2019. *Computational methods for fluid dynamics*. springer.

[33] Kim, Y., Castro, I.P. and Xie, Z.T., 2013. Divergence-free turbulence inflow conditions for large-eddy simulations with incompressible flow solvers. *Computers & fluids*, *84*, pp.56-68.

[34] Pope, S.B., 2001. Turbulent flows. *Measurement Science and Technology*, *12*(11), pp.2020-2021.

[35] Wu, X., 2017. Inflow turbulence generation methods. *Annual Review of Fluid Mechanics*, *49*(1), pp.23-49.

[36] Lund, T.S., Wu, X. and Squires, K.D., 1998. Generation of turbulent inflow data for spatially-developing boundary layer simulations. *Journal of computational physics*, *140*(2), pp.233-258.

[37] Taylor, G.I., 1938. The spectrum of turbulence. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, *164*(919), pp.476-490.

[38] Schau, K.A., Johnson, C., Muller, J. and Oefelein, J.C., 2022. An ensemble Synthetic Eddy Method for accurate treatment of inhomogeneous turbulence. *Computers & Fluids*, *248*, p.105671.

[39] Lozano-Durán, A. and Jiménez, J., 2014. Effect of the computational domain on direct simulations of turbulent channels up to Re_τ= 4200. *Physics of Fluids*, *26*(1).

[40] Bernardini, M., Pirozzoli, S. and Orlandi, P., 2014. Velocity statistics in turbulent channel flow up to. *Journal of Fluid Mechanics*, *742*, pp.171-191.

[41] A visual study of turbulent spots. *Journal of Fluid Mechanics*, *104*, pp.387-405.

[42] Wu, X., 2023. New insights into turbulent spots. *Annual Review of Fluid Mechanics*, *55*(1), pp.45-75.

[43] Sandham, N.D. and Kleiser, L., 1992. The late stages of transition to turbulence in channel flow. *Journal of Fluid Mechanics*, *245*, pp.319-348.

[44] Rabin, S.M.E., Caulfield, C.P. and Kerswell, R.R., 2012. Triggering turbulence efficiently in plane Couette flow. *Journal of Fluid Mechanics*, *712*, pp.244-272.

[45] Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM, et al. Open MPI: Goals, concept, and design of a next-generation MPI implementation. In: Recent advances in parallel virtual machine and message passing interface: 11th European PVM/MPI users' group meeting, Budapest, Hungary, September 19-22, 2004. proceedings 11. Springer; 2004, p. 97–104.

[46] Lee, M. and Moser, R.D., 2015. Direct numerical simulation of turbulent channel flow up to. *Journal of fluid mechanics*, *774*, pp.395-415.

[47] Luchini, P., 2017. Universality of the turbulent velocity profile. *Physical review letters*, *118*(22), p.224501.

[48] Ciarlatani, M.F. and Gorlé, C., 2025. A (co-) kriging multi-fidelity framework for wind loading predictions. *Journal of Building Engineering*, p.112940.