Machine Learning for Complex Fluid Mechanics and Heat Transfer

Diez Sanhueza, R.G.

DOI
10.4233/uuid:dad60122-c7dd-449f-a442-ddb089536648

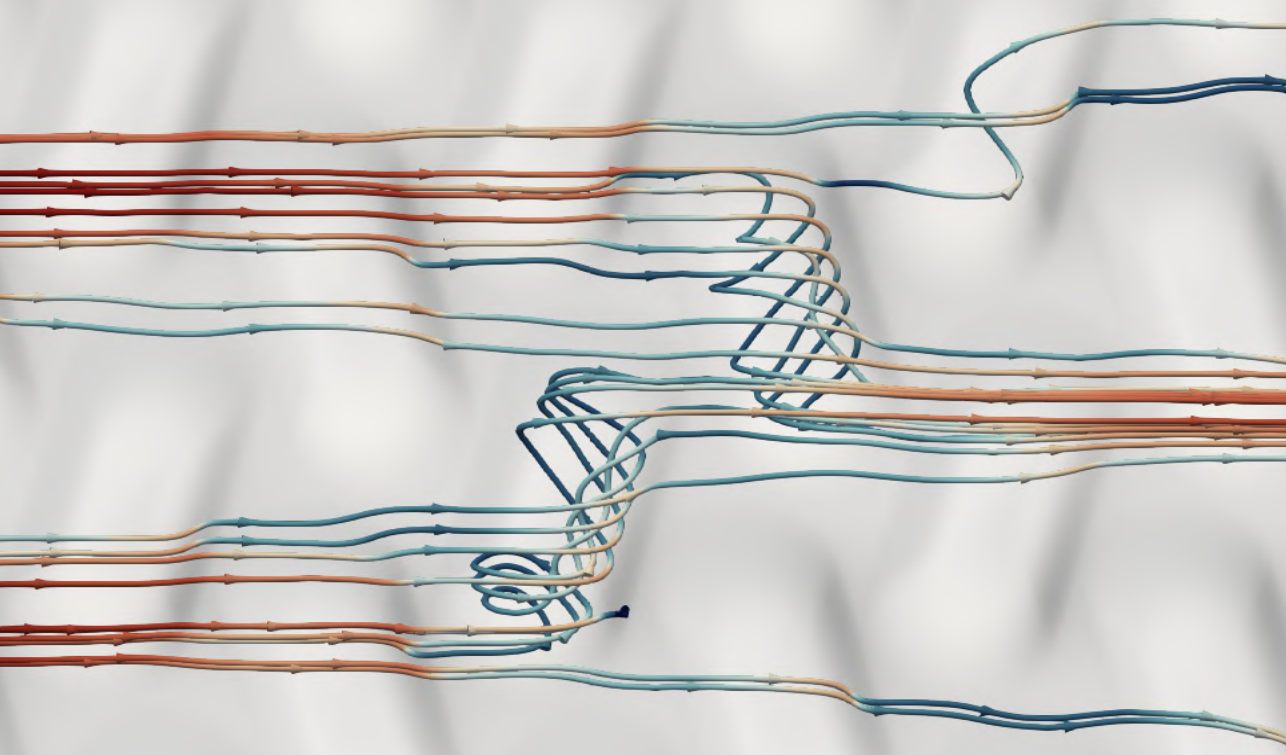
Publication date
2025

Document Version
Final published version

Citation (APA)
Diez Sanhueza, R. G. (2025). *Machine Learning for Complex Fluid Mechanics and Heat Transfer*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:dad60122-c7dd-449f-a442-ddb089536648

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

MACHINE LEARNING FOR COMPLEX FLUID MECHANICS AND HEAT TRANSFER



Rafael DIEZ SANHUEZA

MACHINE LEARNING FOR COMPLEX FLUID MECHANICS AND HEAT TRANSFER

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
3 September 2025 at 17:30 o'clock

by

Rafael Germán DIEZ SANHUEZA

Master of Science in Mechanical Engineering,
Delft University of Technology, Netherlands,
born in Los Angeles, Chile.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. ir. B.J. Boersma	Delft University of Technology, promotor
Dr. ir. J.W.R. Peeters	Delft University of Technology, copromotor
Dr. ir. I. Akkerman	Delft University of Technology, copromotor
Independent members:	
Prof. dr. K. Hooman	Delft University of Technology
Prof. dr. N. Sandham	University of Southampton
Prof. dr. H. Kuerten	Eindhoven University of Technology
Prof. dr. G.D. Weymouth	Delft University of Technology
Dr. ir. W.P. Breugem	Delft University of Technology, reserve member

Keywords: Machine Learning, Turbulence, Heat Transfer, Rough Surfaces, Direct Numerical Simulation, GPU Acceleration, High Performance Computing

Printed by: Print&Bind

Front & Back: Dimple configuration with cross-aligned elements enhancing heat transfer though the creation of spiral vortices.

An electronic version of this dissertation is available at
http://repository.tudelft.nl.

CONTENTS

SUMMARY

Turbulent flows can be found in many industrial applications, and have a profound impact on the drag resistance and heat transfer characteristics of engineering equipment. While the behavior of turbulence is well-documented for canonical flow cases, its behavior under complex conditions is largely unknown. This poses a significant challenge when designing engineering equipment, since it is essential to accurately predict the impact of variable property flows and surface roughness on pressure losses and heat transfer rates. Yet such practical cases are beyond the scope of canonical flow data. During recent years, the combination of advances in machine learning and GPU-accelerated flow solvers has yielded promising results in the fields of turbulence and scalar transport. Using a few modern GPUs and state-of-the-art algorithms, it is now possible to simulate highly complex turbulent flows in short periods of time. This has enabled further progress in the field of machine learning for fluid mechanics, since high-fidelity turbulent flow databases can be easily generated, and new models can be trained to predict increasingly complex flow effects accurately. In this project, multiple challenges are addressed, such as optimizing GPU-accelerated DNS solvers for extreme-scale simulations, creating data-augmented RANS turbulence models for flows with strong variations in their thermophysical properties, and developing machine learning models for turbulent flows past rough surfaces.

Prior to working in GPU-accelerated simulations, the sub-project about machine learning for variable property flows focused on building data-augmented RANS turbulence models using a technique known as FIML (Field Inversion Machine Learning). In this framework, non-linear optimization is first performed to obtain an ideal set of corrections, after which a neural network (or another model) is trained to predict the observed corrections. When working with variable-property flows, several challenges arise, such as not knowing beforehand the local fluid properties due to thermal changes. This makes it difficult, for instance, to calculate the input features for the neural network. The solution created is a feedback loop, where CFD predictions are used to re-calculate the local fluid properties and to update the neural network input features, etc. The results show that the data-augmented RANS model can accurately improve the predictions for unique flow cases, which need unusually high corrections not observed in the training set. Additionally, a weighted relaxation factor methodology is proposed, to ensure convergence of the RANS models after inserting neural network corrections. The final results show that, for the most challenging CFD case identified, our machine learning system is able to reduce the L-infinity error on the velocity profile from 23.4% to 4.0%.

To generate the required amounts of data for machine learning studies regarding complex geometries like roughness, it was necessary to develop a GPU-accelerated DNS solver. This work focuses on the implementation of a parallel tridiagonal solver for extreme-scale simulations, and the creation of a new cross-platform communication library for supercomputers with either AMD or NVIDIA GPUs. In general terms, turbulent

flow simulations in GPUs can be highly efficient, since all operations can be mapped to different GPU threads. However, large-scale data transfers are the main performance bottleneck of GPU-based simulations. While halo exchanges between GPU sub-domains have a minimal impact, the large-scale transpose operations needed for 3D arrays inside Poisson/Helmhotz solvers occupy most of the total running time. Therefore, any transpose operation avoided will drastically improve the running times for the entire DNS solver. By using a parallel tridiagonal solver, it is possible to reduce the number of transposes (for full 3D arrays) by 50% for 2D pencil decompositions inside the Poisson/Helmhotz solvers, and to replace these avoided transposes by simplified operations with a computational cost resembling halo exchanges. Additionally, in this work, a new opportunity to speed up simulations was found, by re-deriving the coefficients of parallel tridiagonal solvers to substantially improve the efficiency and the GPU parallelization of the DNS solver for implicit 1-D diffusion equations. Based on these improvements, it is observed in the results that the efficiency of the DNS solver is substantially improved for extreme-scale simulations in the LUMI and Leonardo supercomputers. Moreover, we show that the entire DNS solver can operate using 2D pencil decompositions without performance degradation compared to most optimal 1D decompositions available for smaller systems. Therefore, the parallel tridiagonal solver enables high-performance in extreme-scale simulations, where 1D decompositions are not feasible.

To enable extreme-scale simulations in AMD GPUs, a new cross-platform communication library was created, named *diezDecomp*. This library is able to achieve high performance working with both CPUs and GPUs in NVIDIA or AMD-based supercomputer. The underlying algorithm corresponds to an advanced implementation that works by directly intersecting the x/y/z bounds of all MPI tasks and scheduling data transfer operations. This allows the implementation of any-to-any transpose operations between mismatched 2D pencil decompositions, with complex communication patterns beyond the scope of traditional all-to-all operations. In extreme-scale simulations, direct x-to-z transposes can improve efficiency while solving for implicit 1D diffusion, but they are not available in existing libraries. Thanks to the flexibility of the *diezDecomp* library, x-to-z transposes can be easily implemented, and the running times of implicit 1D diffusion solvers were improved up to 55% for extreme-scale simulations in the LUMI supercomputer with 1024 GCDs.

The benefits of machine learning to predict the thermal and hydrodynamic behavior of turbulent flows past rough surfaces are explored in Chapter 4. Due to the complexity of this task, a convolutional neural network was used to (independently) scan the input height maps of rough surfaces, and to generate detailed 2-D maps with the local skin friction factors (C_f) and Nusselt numbers (Nu). The proposed neural network is optimized to have linear time complexity while creating 2D maps, instead of quadratic complexity as in naive approaches. The validation study using randomized surfaces with the Fourier spectrum of grit-blasted surfaces shows that machine learning can make accurate 2D predictions for both the local skin friction factors and Nusselt numbers of rough surfaces, with median deviations of 28.43% (C_f) and 6.37% (Nu) respectively. The averaged errors in the predictions for $\overline{C_f}$ and $\overline{Nu}$ were reduced from 24.9% and 13.5% using traditional correlations to only 8.1% and 2.9% thanks to machine learning.

Since the neural network predictions for the thermal behavior of rough surfaces

were particularly promising, a further optimization study is presented in Chapter 5. Here, the objective is to combine the benefits of machine learning and GPU-accelerated simulations to improve convective heat transfer in dimpled-surfaces. Remarkably, it was found that machine learning can find a highly optimized dimpled-surface with a 53% higher Nusselt number using cross-aligned dimples after being trained with only random surfaces (displaying lower thermal performance). After the second iteration of the reinforcement learning loop, it was confirmed that the surface found by machine learning created a flow pattern with helical structures inside the dimples. All other tested parameters had lower thermal performance.

In summary, it is concluded that GPU-based DNS solvers can be optimized to enable extreme-scale simulations with minimal performance degradation. Creating highly flexible communication frameworks, such as the *diezDecomp* library, is also possible while keeping identical running times as traditional libraries. This research project also showcases how machine learning can be an effective tool in fluid mechanics, to predict the behavior of turbulent flows past complex geometries or to account for changes in flows with strong variations in their thermophysical properties.

SAMENVATTING

Turbulente stromingen komen voor in veel industriële toepassingen en hebben aanzienlijke invloed op de warmteoverdrachtskenmerken van technische apparatuur. Hoewel het gedrag van turbulentie goed is gedocumenteerd voor canonieke stromingsgevallen, is het gedrag onder complexe omstandigheden grotendeels onbekend. Dit vormt een aanzienlijke uitdaging bij het ontwerpen van technische systemen, aangezien het essentieel is om nauwkeurig het effect van variabele materiaaleigenschappen en ruwe oppervlaktes op de drukval en warmteoverdracht te voorspellen. Zulke praktische gevallen vallen echter buiten het bereik van canonieke stromingsgegevens. In de afgelopen jaren heeft de vooruitgang in machine learning en GPU-versnelde stromingssimulaties veelbelovende resultaten opgeleverd op het gebied van turbulentie en scalair transport. Met slechts enkele moderne GPU's en geavanceerde algoritmen is het nu mogelijk om zeer complexe turbulente stromingen in korte tijd te simuleren. Dit heeft verdere vooruitgang mogelijk gemaakt in het gebruik van machine learning binnen de stromingsmechanica, aangezien hoogwaardige databases van turbulente stromingen eenvoudig gegenereerd kunnen worden en nieuwe modellen getraind kunnen worden om steeds complexere stromingsfenomenen nauwkeurig te voorspellen.

In dit project worden meerdere uitdagingen aangepakt, zoals het optimaliseren van GPU-versnelde DNS-oplossers voor extreme schaalgroottes, het creëren van data-verrijkte RANS-turbulentiemodellen voor stromingen met sterke variaties in thermofysische eigenschappen en het ontwikkelen van machine learning-modellen voor turbulente stromingen over ruwe oppervlakken.

Voorafgaand aan het werk met GPU-versnelde simulaties richtte het subproject over machine learning voor variabele-eigenschappenstromingen zich op het bouwen van data-verrijkte RANS-turbulentiemodellen met behulp van een techniek genaamd FIML (Field Inversion Machine Learning). Hierbij wordt eerst een niet-lineaire optimalisatie uitgevoerd om een ideale set correcties te verkrijgen, waarna een neurale netwerk (of ander model) wordt getraind om de waargenomen correcties te voorspellen. Bij stromingen met variabele eigenschappen ontstaan meerdere uitdagingen, zoals het vooraf niet weten van lokale vloeistofeigenschappen door thermische veranderingen. Dit bemoeilijkt bijvoorbeeld de berekening van invoerkenmerken voor het neurale netwerk. De oplossing bestaat uit een terugkoppelingslus, waarbij CFD-voorspellingen worden gebruikt om de lokale vloeistofeigenschappen opnieuw te berekenen en de invoerkenmerken voor het neurale netwerk te updaten. De resultaten tonen aan dat het data-verrijkte RANS-model de voorspellingen voor unieke stromingsgevallen nauwkeurig kan verbeteren, zelfs wanneer deze gevallen ongewoon hoge correcties vereisen die niet aanwezig waren in de trainingsdata. Daarnaast wordt een methode met gewogen relaxatiefactor voorgesteld om convergentie van het RANS-model te garanderen na invoeging van de neurale netwerkcorrecties. De eindresultaten tonen aan dat voor het meest uitdagende CFD-geval de L-infinity fout in het snelheidsprofiel kon worden teruggebracht van 23,4% naar 4,0%.

Om de benodigde hoeveelheid data te genereren voor machine learning-studies rond complexe geometrieën zoals wandruwheid, was het nodig een GPU-versnelde DNS-code te ontwikkelen. Dit werk richt zich op de implementatie van een parallelle tridiagonale oplosser voor simulaties op extreme schaal en het maken van een nieuwe cross-platform communicatiebibliotheek voor supercomputers met AMD- of NVIDIA-GPU's. Over het algemeen kunnen turbulente stromingssimulaties op GPU's zeer efficiënt zijn, omdat alle bewerkingen op verschillende GPU-threads kunnen worden gemapt. Echter, grootschalige datatransfers vormen de belangrijkste prestatiebeperking van GPU-gebaseerde simulaties. Hoewel halo-uitwisselingen tussen GPU-subdomeinen een minimale impact hebben, nemen de grootschalige transpositie-operaties die nodig zijn voor 3D-arrays in Poisson-/Helmholtz-oplossers het grootste deel van de totale rekentijd in beslag. Elke vermeden transpositie-operatie leidt daarom tot een drastische verbetering van de rekentijd voor de gehele DNS-code. Met behulp van een parallelle tridiagonale oplosser is het mogelijk om het aantal transposities (voor volledige 3D-arrays) met 50% te verminderen bij 2D pencil-decomposities in Poisson-/Helmholtz-oplossers en deze vervangen door vereenvoudigde bewerkingen met berekeningskosten die lijken op halo-uitwisselingen. Verder werd in dit werk een nieuwe mogelijkheid ontdekt om simulaties te versnellen, door de coëfficiënten van parallelle tridiagonale oplossers opnieuw af te leiden om de efficiëntie en GPU-parallelisatie aanzienlijk te verbeteren voor impliciete 1D-diffusievergelijkingen. Op basis van deze verbeteringen laten de resultaten zien dat de efficiëntie van de DNS-code aanzienlijk is verbeterd voor simulaties op extreme schaal op de supercomputers LUMI en Leonardo. Bovendien tonen we aan dat de volledige DNS-oplosser kan werken met 2D pencil-decomposities zonder prestatienadeel ten opzichte van de meest optimale 1D-decomposities voor kleinere systemen. De parallelle tridiagonale oplosser maakt dus hoge prestaties mogelijk voor simulaties op extreme schaal, waarbij 1D-decomposities niet haalbaar zijn.

Om simulaties op extreme schaal mogelijk te maken op AMD-GPU's is een nieuwe cross-platform communicatiebibliotheek ontwikkeld, genaamd diezDecomp. Deze bibliotheek kan hoge prestaties behalen op zowel CPU's als GPU's in NVIDIA- of AMD-gebaseerde supercomputers. Het onderliggende algoritme is een geavanceerde implementatie die werkt door direct de x/y/z-grenzen van alle MPI-taken te kruisen en gegevensoverdrachten in te plannen. Dit maakt de implementatie van willekeurige transposities mogelijk tussen niet-overeenkomende 2D pencil-decomposities, met complexe communicatiepatronen die buiten het bereik vallen van traditionele all-to-all-operaties. In simulaties op extreme schaal kunnen directe x-naar-z-transposities de efficiëntie verbeteren bij het oplossen van impliciete 1D-diffusie, maar deze zijn niet beschikbaar in bestaande bibliotheken. Dankzij de flexibiliteit van de diezDecomp-bibliotheek kunnen x-naar-z-transposities eenvoudig worden geïmplementeerd en is de rekentijd van impliciete 1D-diffusie-oplossers met maximaal 55% verbeterd in extreme schaal simulaties op de LUMI-supercomputer met 1024 GCD's.

De voordelen van machine learning om het thermische en hydrodynamische gedrag van turbulente stromingen over ruwe oppervlakken te voorspellen worden onderzocht in Hoofdstuk 4. Vanwege de complexiteit van deze taak werd een convolutioneel neurale netwerk gebruikt om (onafhankelijk) de hoogtekaarten van ruwe oppervlakken te scannen en gedetailleerde 2D-kaarten te genereren met lokale wrijvingscoëfficiënten (C_f) en

Nusselt-getallen (Nu). Het voorgestelde neurale netwerk is geoptimaliseerd om lineaire tijdcomplexiteit te hebben bij het creëren van 2D-maps, in plaats van kwadratische complexiteit zoals bij naïeve benaderingen. De validatiestudie met willekeurige oppervlakken gemaakt met behulp van het Fourierspectrum van grit-blasted oppervlakken toont aan dat machine learning nauwkeurige 2D-voorspellingen kan maken voor zowel lokale C_f als Nu, met mediaan afwijkingen van respectievelijk 28,43% (C_f) en 6,37% (Nu). De gemiddelde fouten in de voorspellingen voor $\overline{C_f}$ en $\overline{Nu}$ werden teruggebracht van 24,9% en 13,5% met traditionele correlaties naar slechts 8,1% en 2,9% dankzij machine learning.

Omdat de voorspellingen van het neurale netwerk voor het thermische gedrag van ruwe oppervlakken veelbelovend waren, wordt een verdere optimalisatiestudie gepresenteerd in Hoofdstuk 5. Hier is het doel om de voordelen van machine learning en GPU-versnelde simulaties te combineren om convectieve warmteoverdracht te verbeteren op oppervlakken met dimples. Opmerkelijk is dat machine learning een geoptimaliseerde dimpled-oppervlakte kon vinden met een 53% hoger Nusselt-getal door kruisgewijs uitgelijnde dimples, na getraind te zijn op enkel willekeurige oppervlakken (met lagere thermische prestaties). Na de tweede iteratie van de reinforcement learning-lus werd bevestigd dat het door machine learning gevonden oppervlak een stromingspatroon met helixstructuren binnen de dimples genereerde. Alle andere geteste parameters leverden lagere thermische prestaties op.

Samenvattend kan worden geconcludeerd dat GPU-gebaseerde DNS-codes geoptimaliseerd kunnen worden om simulaties op extreme schaal mogelijk te maken met minimale prestatiedalingen. Het creëren van flexibele communicatieframeworks zoals de diezDecomp-bibliotheek is eveneens mogelijk zonder prestatieverlies ten opzichte van traditionele bibliotheken. Dit onderzoeksproject toont bovendien aan dat machine learning een effectief hulpmiddel kan zijn in stromingsmechanica, om het gedrag van turbulente stromingen over complexe geometrieën te voorspellen of om rekening te houden met variaties in stromingen met sterk wisselende thermofysische eigenschappen.

1

INTRODUCTION

1.1. TURBULENT FLOWS

Turbulent flows can be found in many engineering applications. The presence of turbulence can drastically increase the drag resistance, fuel consumption and heat transfer characteristics of industrial equipment. Despite its relevance, predicting the behavior of turbulence corresponds to one of the most important unsolved problems in physics. While laminar flows can be studied using relatively simple numerical or analytical methods, turbulence corresponds to a chaotic (nonlinear) phenomenon, which can only be resolved though high-resolution unsteady flow simulations. These simulations require massive computational power, since the eddies found in turbulence can reach microscopic sizes for real-world applications. Moreover, the length scale of turbulent eddies becomes smaller for flows at high Reynolds numbers. Mathematically, the grid refinement for turbulent wall-bounded flows scales with $Re_\tau^{11/4}$, where Re_τ is the friction Reynolds number. Simplifying the physics of turbulent flows is difficult, because turbulence can be seen as an energy cascade, where large eddies drive the generation of smaller eddies, with sizes varying across many orders of magnitude. All these eddies are intricately interconnected though the non-linear Navier-Stokes equations, and they cannot be analyzed separately.

During the last decades, the amount of computing power available in the world has increased exponentially. This has enabled the simulation of flows at significantly higher Reynolds numbers, starting from the work of (Orszag & Patterson, 1972). The recent advances in GPU technology have enabled a new breakthrough in flow simulations. Today, a single GPU in a desktop computer can be as fast as an entire supercomputer from a decade ago. This has increased the feasibility of studying turbulent flows under complex conditions, and the generation of rich databases for model assimilation. However, predicting turbulence is a complex task. Due to their nonlinear nature, mathematical models for turbulence must be carefully designed to achieve numerical robustness, and extrapolating to new scenarios might be challenging. Moreover, GPUs have different characteristics than CPU architectures, and different algorithms are necessary to fully

utilize their capacity. Therefore, modern research focuses both on the development of GPU-accelerated fluid solvers, and the generation of new physical models using turbulent flow databases. Within this context, machine learning is a promising technique for model assimilation, since neural networks and other data-driven approaches can be trained using the rich turbulent flow databases generated by GPUs. Moreover, research shows that high-fidelity turbulent flow data usually contains a large variety of complex non-linear effects that are beyond the scope of traditional (simplified) models. Thus, using machine learning is necessary to develop accurate models that fully take into account the complex phenomena observed in practice.

In the classical literature, the behavior of turbulence is mainly well-documented for canonical cases, such as flows around simple geometries, or flows with limited variations in their thermophysical properties. However, many engineering applications involve complex geometries (Busse et al., 2015; Flack & Schultz, 2010; Peeters & Sandham, 2019), or flows with strong property gradients (Pecnik & Patel, 2017). Due to the increased amount of computing power available, it is now more feasible to study these types of flows than before. In the case of flows with strong variations in their thermophysical properties, traditional RANS turbulence models can be inaccurate even for smooth channels, unless new correction terms are introduced (Rodriguez et al., 2018).

For flows around complex geometries, the main challenge is that traditional RANS turbulence models have been calibrated using simple cases (pipelines, free-shear layers, etc.), and their differential equations may be inaccurate for non-trivial geometries. Moreover, if the walls are hydrodynamically rough, another issue is the estimation of the equivalent Nikuradse sand-grain roughness height (k_s^+) needed by wall functions in RANS models. Mathematically, k_s^+ depends on the drag resistance generated by the rough surface, and thus converting 2-D surface features into an accurate k_s^+ estimation is a great challenge. When heat transfer is considered, similar issues are encountered to estimate the Nusselt number of rough surfaces. Additionally, thermal equations in RANS models require closure models for the turbulent Prandtl number, which is only an approximation, and it may change through the geometry of the flow. Due to the previous circumstances, innovative solutions are needed to create improved models for fluid mechanics.

1.2. MACHINE LEARNING

In order to build new predictive models for fluid mechanics, one of the main challenges is to process large amounts of data, and to create models with enough flexibility to account for a large variety of complex flow cases. The behavior of turbulence rarely follows simple trends, and the observed behavior (in non-trivial cases) is usually caused by nonlinear interactions between many factors. Due to these reasons, machine learning is an ideal technique to build new models for fluid mechanics. Machine learning has achieved state-of-the-art results across many fields of science in the last decade. Moreover, the recent advances in GPU technology have greatly increased the availability of computing power for scientific applications. In the field of machine learning for fluid mechanics, different approaches have been studied. For instance, (Ling et al., 2016) developed a deep neural network with embedded Galilean invariance to predict anisotropic Reynolds stress tensors for various flow cases. To improve existing RANS models, (Parish & Duraisamy, 2016) proposed a framework known as field inversion machine learning (FIML). In this

technique, field inversion is first performed to obtain an ideal set of corrections that minimize the discrepancy between the target RANS model and the DNS data. Then, a neural network is trained to replicate the corrections identified by field inversion, and the results are injected in a data-augmented CFD solver. The FIML technique has been applied successfully to various turbulence models and flow geometries (Fang & He, 2024; Singh, Duraisamy, & Zhang, 2017; Singh, Medida, & Duraisamy, 2017). Another interesting technique is to use machine learning to identify new algebraic terms for existing nonlinear models (Brunton et al., 2016; Weatheritt & Sandberg, 2017). The main benefit of this approach is that the optimization process (replacing field inversion) already yields a new model, which can be tested. However, the quality of the results depends on the compatibility of the proposed algebraic terms with the original RANS model, and the complexity of the dataset studied. In the case of rough surfaces, multiple authors have used machine learning to determine which surface metrics are best to predict the equivalent Nikuradse sand-grain roughness height (k_s^+) (Jouybari et al., 2021; Ma et al., 2023). Further applications of machine learning in fluid mechanics involve topics, such as: 3-D flow reconstruction, reduced order modelling, Proper Orthogonal Decomposition (POD), etc. A detailed review of different types of applications can be found in (Brunton et al., 2020; Sharma et al., 2023).

1.3. OBJECTIVES

The main goals of this thesis are: (1) to create new machine learning models for complex applications involving fluid mechanics or heat transfer, and (2) to optimize GPU-accelerated DNS solvers for extreme-scale simulations. The combination of machine learning and GPU simulations is ideal, since GPU-based DNS solvers make it feasible to generate rich databases with high-fidelity data regarding complex flows, whereas machine learning can extract patterns from this data and create low-cost models for evaluation. Given this context, different lines of work are considered, which are divided into the following broad categories:

- Improve the performance and scalability of existing DNS solvers for extreme-scale simulations in GPU-based supercomputers (Leonardo, Snellius, LUMI). This is accomplished though the implementation of a parallel tridiagonal solver with an additional 2-D pencil decomposition for simulations at very high Reynolds numbers.

- Create a new high-performance communication library for extreme-scale DNS studies in supercomputers, which is cross-platform compatible with CPUs and either NVIDIA or AMD GPUs.

- Develop a FIML framework for data-augmented RANS turbulence models, which is able of handle flows with strong variations in their thermophysical properties. Also, build a robust stabilization methodology to avoid spurious oscillations in such flows.

- Create a novel convolutional neural network architecture, with optimized time and space complexity, which can process the height maps of rough surfaces directly, and

perform detailed 2-D predictions about their local Nusselt numbers or skin friction factors. This technique avoids relying on traditional surface metrics (skewness, effective slope, etc.), and it can process much more complex cases.

- Build a novel machine learning framework for the optimization of convective heat transfer in dimpled surface configurations. The methodology corresponds to a reinforcement learning framework that is trained with the results of various DNS cases, and it uses the previous convolutional neural network to make predictions.

1.4. THESIS OUTLINE

Based on the previous objectives, the thesis chapters are organized as follows:

- **Chapter 2** presents the general numerical methods used in this work, together with the improvements made for DNS solvers running extreme-scale simulations in GPU-based supercomputers. The new communication library for supercomputer simulations is also discussed in this chapter.
- **Chapter 3** describes the generation of data-augmented RANS turbulence models for flows with strong variations in their thermophysical properties, using a FIML framework.
- **Chapter 4** explains the details of the new convolutional neural network to predict detailed 2-D maps with the local Nusselt numbers and skin friction factors generated by rough surfaces.
- **Chapter 5** is dedicated to the novel machine learning framework to optimize dimpled surfaces for convective heat transfer enhancement, using the convolutional neural network described in Chapter 4.
- **Chapter 6** summarizes the current thesis, and presents the conclusions of the study.

REFERENCES

Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics, 52*(Volume 52, 2020), 477–508. https://doi.org/https://doi.org/10.1146/annurev-fluid-010719-060214

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences, 113*(15), 3932–3937. https://doi.org/10.1073/pnas.1517384113

Busse, A., Lützner, M., & Sandham, N. D. (2015). Direct numerical simulation of turbulent flow over a rough surface based on a surface scan. *Computers & Fluids, 116*, 129–147. https://doi.org/https://doi.org/10.1016/j.compfluid.2015.04.008

Fang, L., & He, P. (2024). Field inversion machine learning augmented turbulence modeling for time-accurate unsteady flow. *Physics of Fluids, 36*(5), 055117. https://doi.org/10.1063/5.0207704

Flack, K. A., & Schultz, M. P. (2010). Review of Hydraulic Roughness Scales in the Fully Rough Regime [041203]. *Journal of Fluids Engineering*, *132*(4). https://doi.org/10.1115/1.4001492

Jouybari, M. A., Yuan, J., Brereton, G. J., & Murillo, M. S. (2021). Data-driven prediction of the equivalent sand-grain height in rough-wall turbulent flows. *Journal of Fluid Mechanics*, *912*, A8. https://doi.org/10.1017/jfm.2020.1085

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, *807*, 155–166.

Ma, H., Li, Y., Yang, X., & Ye, L. (2023). Data-driven prediction of the equivalent sand-grain roughness. *Scientific Reports*, *13*(1), 19108. https://doi.org/10.1038/s41598-023-46564-4

Orszag, S. A., & Patterson, G. S. (1972). Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett.*, *28*, 76–79. https://doi.org/10.1103/PhysRevLett.28.76

Parish, E., & Duraisamy, K. (2016). A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, *305*, 758–774.

Pecnik, R., & Patel, A. (2017). Scaling and modelling of turbulence in variable property channel flows. *Journal of Fluid Mechanics*, *823*, R1.

Peeters, J., & Sandham, N. (2019). Turbulent heat transfer in channels with irregular roughness. *International Journal of Heat and Mass Transfer*, *138*, 454–467. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2019.04.013

Rodriguez, G. J. O., Patel, A., S., R. D., & Pecnik, R. (2018). Turbulence modelling for flows with strong variations in thermo-physical properties. *International Journal of Heat and Fluid Flow*, *73*, 114–123.

Sharma, P., Chung, W. T., Akoush, B., & Ihme, M. (2023). A review of physics-informed machine learning in fluid mechanics. *Energies*, *16*(5). https://doi.org/10.3390/en16052343

Singh, A. P., Duraisamy, K., & Zhang, Z. J. (2017, January). Augmentation of turbulence models using field inversion and machine learning. American Institute of Aeronautics; Astronautics.

Singh, A. P., Medida, S., & Duraisamy, K. (2017). Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, *55*(7), 2215–2227.

Weatheritt, J., & Sandberg, R. (2017). The development of algebraic stress models using a novel evolutionary algorithm. *International Journal of Heat and Fluid Flow*, *68*, 298–318. https://doi.org/https://doi.org/10.1016/j.ijheatfluidflow.2017.09.017

2

OPTIMIZED DNS SOLVER FOR EXTREME-SCALE CALCULATIONS

We present a computational method for extreme-scale simulations of incompressible turbulent wall flows at high Reynolds numbers. The numerical algorithm extends a popular method for solving second-order finite differences Poisson/Helmholtz equations using a pencil-distributed parallel tridiagonal solver to improve computational performance at scale. The benefits of this approach were investigated for high-Reynolds-number turbulent channel flow simulations, with up to about 80 billion grid points and 1024 GPUs on the European flagship supercomputers Leonardo and LUMI. An additional GPU porting effort of the entire solver had to be undertaken for the latter. Our results confirm that, while 1D domain decompositions are favorable for smaller systems, they become inefficient or even impossible at large scales. This restriction is relaxed by adopting a pencil-distributed approach. The results show that, at scale, the revised Poisson solver is about twice as fast as the baseline approach with the full-transpose algorithm for 2D domain decompositions. Strong and weak scalability tests show that the performance gains are due to the lower communication footprint. Additionally, to secure high performance when solving for wall-normal implicit diffusion, we propose a reworked flavor of parallel cyclic reduction (PCR) that is split into pre-processing and runtime steps. During pre-processing, small sub-arrays with independent 1D coefficients are computed by parallel GPU threads, without any global GPU communication. Then, at runtime, the reworked PCR enables a fast solution of implicit 1D diffusion without computational overhead. Our results show that the entire numerical solver, coupled with the PCR algorithm, enables extreme-scale simulations with 2D pencil decompositions, which do not suffer performance losses even when compared to the best 1D slab configurations available for smaller systems.

This chapter has been published in Computer Physics Communications, vol. 316, p. 109811, 2025.

2.1. INTRODUCTION

Turbulent flows at high Reynolds numbers are among the most complex and prevalent problems in engineering and physics. While numerous flows at low Reynolds numbers may be studied using simple analytical or numerical models, many flows found in nature and industry operate at high Reynolds numbers in a turbulent regime. These flows exhibit complex behavior, which is difficult to predict using existing correlations or upscaled models. Fundamental understanding of turbulence that can improve engineering models requires direct numerical simulations (DNS), where the chaotic and multi-scale flow dynamics are fully resolved up to the smallest temporal and spatial scales. While this has a large computational cost, the exponential growth in computing power during the last decades, along with the development of efficient numerical methods, have enabled the simulation of flows at increasingly high Reynolds numbers. Indeed, following the developments since the first DNS of isotropic turbulence by (Orszag & Patterson, 1972), it is now possible to perform large-scale simulations with trillions of grid points in modern supercomputers (Ishihara et al., 2009; Yeung & Ravikumar, 2020). In the context of this thesis, large-scale DNS is necessary to study both turbulent flows past rough surfaces and variable-property flows. In the case of rough surfaces, large DNS grid sizes are required to fully resolve the flow patterns near irregular surfaces; even at moderate Reynolds numbers. For variable-property flows, the simulation time steps are usually smaller than incompressible flows with similar grid sizes, and thus it is ideal to speed-up the simulations using the well-optimized parallelization techniques found in large-scale DNS solvers.

We are currently experiencing another breakthrough, thanks to the proliferation of general-purpose GPU-based supercomputers (Nickolls & Dally, 2010; Reed et al., 2023). GPUs are known to perform well in tasks that only require simple arithmetic operations or RAM access patterns ("CUDA Fortran for Scientists and Engineers", 2014), which are common in computational fluid dynamics (CFD). They have much higher throughput than CPUs by allowing many parallel threads to perform the same operation per clock cycle. Thus, when successfully ported, GPU-accelerated solvers can easily outperform multi-core CPU solvers (Y. Kim et al., 2023; Salvadore et al., 2013), enabling the numerical solution to complex problems at much lower costs. However, large-scale CFD problems need to operate at scale, in a distributed-memory paradigm where the data is distributed among many GPUs. This introduces new challenges, as many-GPU systems are more prone to feature performance bottlenecks associated with intra and internode communication, which may require adjustments in the numerical algorithm. Let us consider the current European pre-exascale flagship supercomputers Leonardo and LUMI. A summary of the GPU specifications for both supercomputers can be found in Table 2.1. In the case of LUMI, it is important to highlight that each MI250x is split into two Graphics Compute Dies (GCDs)[1] (*AMD Instinct MI250X Accelerators*, 2025), each with roughly similar characteristics as a NVIDIA A100 GPU in terms of memory and processing power. The internal memory bandwidths for both AMD and NVIDIA GPUs are much faster than intra- and inter-node communication. Therefore, algorithms that minimize device-to-device communication can be optimal, even if they slightly increase the number of arithmetic

[1] In this work, we consider each GCD in LUMI to be an independent GPU (unless explicitly noted), since they are exposed to the user's software as separate devices.

operations or HBM (High Memory Bandwidth) usage per device. Additionally, opportunities for code optimization can be found by selecting algorithms that replace inter-node communication with faster intra-node data transfers.

Table 2.1: Specifications for the GPU nodes in the Leonardo and LUMI supercomputers (*AMD Instinct MI250X Accelerators*, 2025; De Sensi et al., 2024; *Nvidia A100 Tensor Core GPU: Unprecedented Acceleration at Every Scale*, 2024; Schieffer et al., 2025). The data formally corresponds to the Booster partition in Leonardo, and the LUMI-G nodes for LUMI. The abbreviation "Gb" refers to Gigabit, whereas "GB"/"TB" corresponds to Gigabyte/Terabyte.

	Leonardo	LUMI
GPU model	NVIDIA A100	AMD MI250X
Number of devices	4 GPUs	8 GCDs (2 per GPU)
Inter-node bandwidth	100 Gb/s	100 Gb/s[†]
Intra-node bandwidth	800 Gb/s	400 – 1600 Gb/s[†]
Internal memory bandwidth	1.6 TB/s	1.6 TB/s[†]
HBM	64 GB	64 GB[†]
FP64 Peak Performance	9.7 TFLOPs 19.5 TFLOPs with Tensor Cores	23.95 TFLOPs[†]

[†] Data for each GCD in LUMI.

Several works have performed large-scale turbulent flow simulations using multi-GPU configurations. Compressible flow solvers, for instance, contain many fully explicit calculations that can be readily parallelized using GPUs. The DNS solver *STREAmS* (Bernardini et al., 2021) can use multi-GPU systems to simulate compressible wall-bounded flows, taking into account complex effects such as shock-wave interactions. In *URANOS* (De Vanna et al., 2023), a compressible flow solver is developed for large-scale simulations using various modelling frameworks, and several possible choices for the discretization schemes. In incompressible flow solvers, GPU porting for distributed-memory calculations at scale faces an extra challenge. Typically, a major performance bottleneck is solving a large linear system associated with the pressure Poisson equation to ensure incompressibility. Nevertheless, several recent works have shown great progress in the development of multi-GPU solvers. Focusing on spectral or finite-difference approaches, an example is the AFiD-GPU code (Zhu et al., 2018) for large-scale simulations of wall-bounded flows using multi-GPU (or multi-CPU) configurations. Another example is the CaNS code, which is used in the present study (Costa, 2018; Costa et al., 2021). CaNS is an incompressible DNS solver, which is compatible with various types of boundary conditions for canonical flow cases in rectangular grids, such as isotropic turbulence or wall-bounded flows. This solver is compatible with both multi-GPU and multi-core CPU architectures, and has been recently re-ported to GPUs porting using OpenACC and the hardware-adaptive *cuDecomp* library for GPU communications at scale (Romero et al., 2022). This library allows pencil-distributed solvers that require collective transpose operations to perform runtime autotuning and determine the optimal domain decomposition

and GPU communication backend. A simple FFT-based finite-differences numerical solver like CaNS requires two types of communication operations: halo exchanges and transposes. Halo exchanges are relatively simple, standard operations, where each task exchanges boundary values with its neighbors. Transposes are more expensive *all-to-all* collective operations, where 3D data of a field is redistributed among MPI tasks such that all cells aligned in a specific dimension are local to a single process/task. This is important while performing, for instance, fast Fourier-based transformations in spectral Poisson/Helmholtz solvers, since FFT algorithms require frequent access to the arrays being transformed. Naturally, all transpose operations involving 3D arrays are expensive, and often the major performance bottleneck.

Second-order FFT-based finite-difference solvers such as those used in AFiD and CaNS require performing FFTs along two directions, and the solution of a resulting tridiagonal system along the other direction, which is typically the wall-normal one in the case of wall-bounded flows with one inhomogeneous direction. The solution of the tridiagonal system has been typically performed using transpose operations, such that the whole system is local to each task and serially solved. However, there is possible room for improvement here by exploiting a parallel tridiagonal solver that avoids this collective operation. Notably, an interesting approach was presented by (László et al., 2016) and exploited in (Gong et al., 2022; K.-H. Kim et al., 2021; Yang, Kang, et al., 2023; Yang, Oh, et al., 2023), which showed compelling performance gains at scale. In short, this method uses a hybrid Thomas–parallel cyclic reduction (PCR) algorithm that effectively converts the tridiagonal system to be solved in parallel into a series of smaller systems that can be solved independently, coupled to a smaller problem to be solved collectively for the first and last unknowns of each small system (Gong et al., 2022).

Most works exploring PCR in this context have adopted a 1D parallelization (K.-H. Kim et al., 2021; Yang, Oh, et al., 2023), with slabs parallel to directions of FFT-based synthesis. This is efficient and was proven to work up to a certain scale. However, as the flow Reynolds number increases, it becomes impossible to resort to a 1D parallelization. As an example, Figure 2.1 presents the total memory requirement in a DNS solver (CaNS), to simulate turbulent channel flows at increasing friction Reynolds number (Re_τ), as well as the total size of a single $n_x \times n_y$ wall-parallel slice. Expectedly, as Re_τ increases, in addition to stricter time steps restrictions, the number of grid cells increases roughly as $N_{x,y} \propto Re_\tau$ in the streamwise (x) and spanwise (y) directions of the channel flow, and $N_z \propto Re_\tau^{3/4}$ in the wall-normal direction (z) (Pirozzoli & Orlandi, 2021; Pope, 2001). Hence, as the Reynolds number increases, the thickness of a wall-parallel slab that fits a fixed amount of grid points (e.g., dictated by the RAM constrains of a GPU or CPU device) becomes ever thinner, until it becomes impossible to decompose the domain further. This is particularly problematic in wall-bounded turbulence, where the number of grid points along the wall-parallel directions should be larger than in the wall-normal one (Pirozzoli & Orlandi, 2021). Yet, the same is bound to happen in other turbulent flows (e.g., homogeneous isotropic turbulence) at sufficiently high Reynolds number.

Consequently, even with the ever-increasing memory capacity of GPUs, for DNS of high Reynolds number flows with this type of approach, one may be bound to adopt a 2D pencil-like domain decomposition. Leveraging a less communication-intensive approach for solving the Poisson equation, while retaining a pencil-distributed decomposition, is

precisely the motivation of the present work. One alternative within this context is to replace serial TDMA (tridiagonal matrix algorithms) by parallel methods, with a lower communication footprint. We develop such an approach based on a PCR-TDMA method (named P-TDMA hereafter), and test it on the CaNS solver with a focus on many-GPU calculations at scale. To run our solver on an AMD-based system like LUMI, an additional porting effort was undertaken, which we will also describe here. We report the revised solver's performance and scalability for large-scale DNS of turbulent channel flow at high Reynolds number. Moreover, we show that particular care should be taken for wall-normal implicit diffusion, to secure high performance at scale. Our results show an almost 2× speedup at scale for the Poisson solver with 2D decomposition, which significantly improves the overall solver performance on large-scale GPU-based supercomputers.

This chapter is organized as follows. Section 2.2 describes the governing equations, the discretization scheme, and the implementation of the Poisson/Helmholtz solver, including the cross-platform effort to run on AMD-based systems. Then Section 2.3 discusses the study and the scalability benchmarks. Finally, Section 2.4 presents the conclusions.

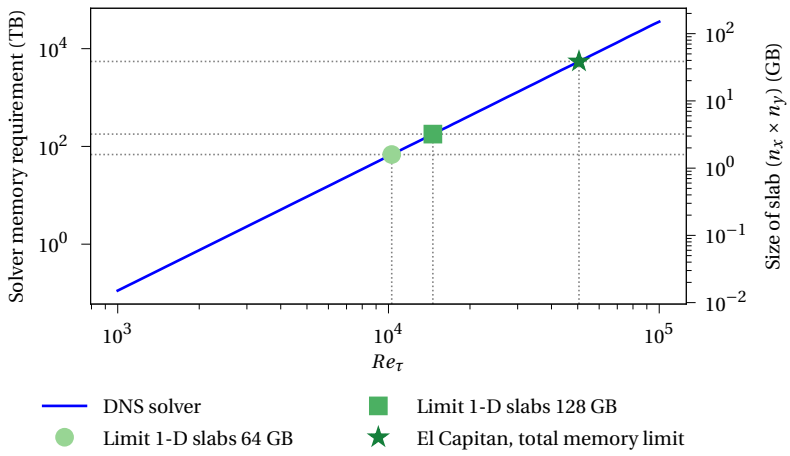


Figure 2.1: Estimates of the total memory requirements for double-precision DNS runs of turbulent channel flows on a $L_x \times L_y \times L_z = 12.8 \times 6.4 \times 2$ domain at different friction Reynolds numbers (Re_τ), as well as the size of a single slice $n_x \times n_y$ of the DNS domain in HBM for all points in the streamwise (x) and spanwise (y) directions. The calculations are performed using the domain partition algorithms of the DNS solver CaNS (Costa et al., 2021) and the parallel decomposition library *cuDecomp* (Romero et al., 2022). The symbols denote the limits where a 1D slab decomposition becomes impossible for GPUs with 64 GB and 128 GB of memory, respectively, which is typical of current high-end HPC GPUs, along with the total GPU memory of the largest supercomputer as of 2025: El Capitan ("TOP500 list - november 2024", 2024). This marks an upper bound of the maximum Reynolds number that could be investigated with current computational resources: $\mathrm{Re}_\tau \approx 50{,}000$.

2.2. METHODOLOGY

2.2.1. GOVERNING EQUATIONS AND NUMERICAL DISCRETIZATION

The current numerical framework solves the incompressible Navier-Stokes equations,

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u}, \tag{2.2}$$

where $\mathbf{u}$ and p correspond to the fluid velocity vector and the pressure scaled by the fluid density; ν is the fluid kinematic viscosity. The numerical scheme is based on an incremental pressure correction scheme or fractional-step method (Chorin, 1967; J. Kim & Moin, 1985), where a prediction velocity $\mathbf{u}^*$ is first calculated by integrating the momentum equation in time, and continuity is imposed using a correction pressure Φ, which is obtained from the solution of a Poisson equation. The equations were solved on a rectangular box using a structured Cartesian grid with a staggered (MAC) arrangement for the velocity and pressure grid cells. As per the restriction of the Poisson solver, we employ uniform spacing along two Cartesian directions (x,y) and non-uniform spacing in the third spatial direction (z). Second-order finite differences are used for spatial discretization. This has several advantages with respect to higher-order methods, such as being computationally efficient while still enabling simulations with similar fidelity as spectral discretization methods in practice (Moin & Verzicco, 2016), and being flexible and easily extended with numerical techniques for handling complex geometries like the immersed-boundary method (Breugem & Boersma, 2005; Fadlun et al., 2000; Uhlmann, 2005), or multi-fluid flows (Tryggvason et al., 2011). Finally, Wray's low-storage Runge-Kutta scheme (Wray, 1986) is used for temporal discretization. The numerical scheme is presented below in semi-discrete form:

$$\mathbf{u}^* = \mathbf{u}^k + \Delta t \left(\alpha_k \left(\mathscr{A}\mathbf{u}^k + \nu \mathscr{L}\mathbf{u}^k \right) + \beta_k \left(\mathscr{A}\mathbf{u}^{k-1} + \nu \mathscr{L}\mathbf{u}^{k-1} \right) - \gamma_k \mathscr{G} p^{k-1/2} \right), \tag{2.3}$$

$$\mathscr{L}\Phi = \frac{\mathscr{D}\mathbf{u}^*}{\gamma_k \Delta t}, \tag{2.4}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^* - \gamma_k \Delta t \mathscr{G}\Phi, \tag{2.5}$$

$$p^{k+1/2} = p^{k-1/2} + \Phi. \tag{2.6}$$

where α, β, and γ refer to the coefficients of the RK3 scheme, which are given by: $\alpha = \{8/15,\ 5/12,\ 3/4\}$, $\beta = \{0,\ -17/60,\ -5/12\}$ and $\gamma = \alpha + \beta$; the index k refers to the RK3 sub-iteration index $k = \{0,1,2\}$, and Δt is the time step. For flows at very low Reynolds numbers, or highly refined grids, the time step size Δt can be prohibitively small if diffusive terms are integrated in time explicitly. In such cases, it may be preferable to perform an implicit discretization of the diffusion terms at the cost of solving an extra Helmholtz equation per velocity component, as illustrated below:

$$\mathbf{u}^{**} = \mathbf{u}^k + \Delta t \left(\alpha_k \mathscr{A}\mathbf{u}^k + \beta_k \mathscr{A}\mathbf{u}^{k-1} + \gamma_k \left(-\mathscr{G} p^{k-1/2} + \nu \mathscr{L}\mathbf{u}^k \right) \right), \tag{2.7}$$

$$\mathbf{u}^* - \gamma_k \frac{\nu \Delta t}{2} \mathscr{L}\mathbf{u}^* = \mathbf{u}^{**} - \gamma_k \frac{\nu \Delta t}{2} \mathscr{L}\mathbf{u}^k, \tag{2.8}$$

$$\mathscr{L}\Phi = \frac{\mathscr{D}\mathbf{u}^*}{\gamma_k \Delta t}, \tag{2.4}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^* - \gamma_k \Delta t \mathscr{G}\Phi, \tag{2.9}$$

$$p^{k+1/2} = p^{k-1/2} + \Phi - \gamma_k \frac{\nu \Delta t}{2} \mathscr{L}\Phi. \tag{2.10}$$

Note that, indeed, Eq. (2.8) is a Helmholtz equation for the prediction velocity $\mathbf{u}^*$, which is implicit in the three spatial directions (x,y,z). While this equation can be solved efficiently using the fast direct methods presented in Section 2.2.2, the computational overhead is still considerable. Fortunately, in many cases, the time step constraints for Δt are due to fine grid spacing only along the non-uniform grid direction (here, z). This is particularly true for wall-bounded flows with one inhomogeneous direction, such as pipes or channels, which require fine grid spacing near the walls (Costa, 2018; Zhu et al., 2018). In these cases, one can discretize only the wall-normal diffusion term implicitly, and replace Eq. (2.8) with a one-dimensional system per velocity component:

$$\mathbf{u}^* - \gamma_k \frac{\nu \Delta t}{2} \mathscr{L}_z\mathbf{u}^* = \mathbf{u}^{**} - \gamma_k \frac{\nu \Delta t}{2} \mathscr{L}_z\mathbf{u}^k, \tag{2.11}$$

where $\mathscr{L}_z$ denotes the discrete Laplacian term associated with the z direction. This is numerically much cheaper, as the second-order finite-difference discretization of this equation requires the solution of a simple tridiagonal system.

2.2.2. NUMERICAL SOLUTION OF THE POISSON/HELMHOLTZ EQUATION

FOURIER-BASED SYNTHESIS

The solution of the Poisson equation for the pressure comprises some of the numerical algorithm's most computation and communication-intensive steps. Here, Eqs. (2.4) and (2.8) are solved using the method of eigenfunctions expansions, which allows for fast, direct solutions by leveraging the FFT algorithm (Costa, 2018; Schumann & Sweet, 1988). After performing a Fourier-based synthesis of the Poisson/Helmholtz equation along directions x and y, the following system of tridiagonal equations can be obtained along the non-uniform grid direction z for a grid cell with index i, j, k:

$$\left(\lambda_i/\Delta x^2 + \lambda_j/\Delta y^2\right)\tilde{\Phi}_{i,j,k} + \left(\eta_{k-1}\tilde{\Phi}_{i,j,k-1} + \eta_k\tilde{\Phi}_{i,j,k} + \eta_{k+1}\tilde{\Phi}_{i,j,k+1}\right) = \tilde{f}_{i,j,k}, \tag{2.12}$$

where the tilde ($\tilde{}$) denotes two successive discrete Fourier-based (i.e., Fourier/cosine/sine) transforms applied to a variable along the x (index i) and y (index j) directions. Note that each (i, j) pair corresponds to a tridiagonal system along the non-uniform direction (z, index k). The coefficients λ_i and λ_j are the second-order accurate eigenvalues (or modified wavenumbers); see, e.g., (Schumann & Sweet, 1988); Δx and Δy correspond to the uniform grid spacing in the x and y directions, whereas the set of coefficients η represent the finite-different discretization of the $\mathscr{L}_z$ operator along z. While the method of eigenfunctions expansions aims at exploiting the FFT algorithm, it still allows for multiple combinations of boundary conditions representative of different classes of canonical

turbulent flows, from isotropic turbulence to several boundary-free and wall-bounded shear flows.

After obtaining $\tilde{\Phi}_{i,j,k}$ from Eq. (2.12), the final solution ($\Phi_{i,j,k}$) is easily computed from the inverse Fourier-based synthesis. The numerical methods and algorithms to solve Eq. (2.12) are the key parts of the present work and will be discussed next in Section 2.2.2. Finally, there are noteworthy nuances in implementing fast real-to-complex/complex-to-real (Fourier) and real-to-real (sine/cosine) transforms on GPUs in a unified framework, which we describe in A.1.

ORIGINAL DISTRIBUTED-MEMORY SOLUTION

The numerical solution of the second-order finite-difference Poisson/Helmholtz equations in rectangular grids is solved using FFT-based methods in a distributed-memory setting. The original approach used to solve this problem is described below[2], and illustrated in Figure 2.2. The following steps are taken:

1. Compute right-hand side term $d_{i,j,k}$ of eq. (2.12) in z-aligned pencils.
2. Transpose data to x-aligned pencils and perform $N_y N_z$ Fourier-based transforms along x.
3. Transpose data to y-aligned pencils and perform $N_x N_z$ Fourier-based transforms along y.
4. Transpose data to z-aligned pencils and solve the resulting $N_x N_y$ tridiagonal systems of equations along z.
5. Perform the reciprocate transpose operation as in step 4.
6. Perform the reciprocate inverse transforms and transpose operation as in step 3.
7. Perform the reciprocate inverse transforms and transpose operation as in step 2, to obtain the final solution in z-aligned pencils.

Here, $N_{x/y/z}$ are the local number of grid cells along $x/y/z$ during the different steps of the algorithm for each MPI task. The transpose operations are an *all-to-all* collective, which may be very expensive. Within this approach, the FFT-based transforms and solution of the tridiagonal systems can be trivially mapped to different parallel (GPU) threads. Note that, whenever a 1D slab-like parallelization is possible, some of the transpose operations shown above would turn into a no-op, making it often desirable. In this regard, the best-performing slab configurations are those partitioned along y. This is convenient, since each GPU can perform Fourier transformations along the x-direction, and solve tridiagonal systems of equations along the z-direction, without performing additional collective operations. Even with 1D implicit diffusion, only one pair of transposes is required per step: the $x \leftrightarrow y$ transposes shown in Figure 2.2.

[2] It is important to note that, while the code CaNS allows for an arbitrary default pencil orientation (i.e., outside the pressure solver), we start from Z-aligned pencils since this minimizes the number of collective communications when solving the momentum equation with z-implicit diffusion. Starting from x-aligned pencils would avoid transpose operations in the Poisson solver, but many additional transpose operations would be required for inverting a tridiagonal system per velocity component.

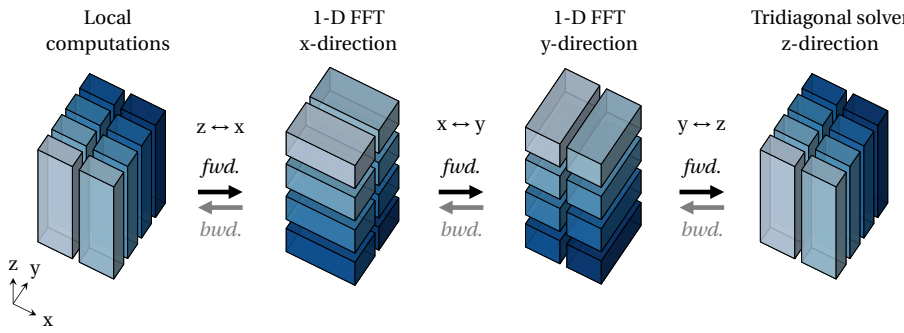


Figure 2.2: Schematic representation of a FFT-based linear solver using a 2D pencil decomposition for the DNS domain. The black arrows indicate the global transpose operations for the data stored among different MPI tasks. The color of each 2D pencil represents a different MPI task. Forward (*fwd.*) operations are first performed from left to right following the direction of the black arrows. Then, all transpose operations are reversed, and inverse/backward (*bwd.*) Fourier-based transforms are performed, as indicated by the gray arrows. Note that the first transpose operation is often implemented as two consecutive transposes: $z \rightarrow y$ and $y \rightarrow x$.

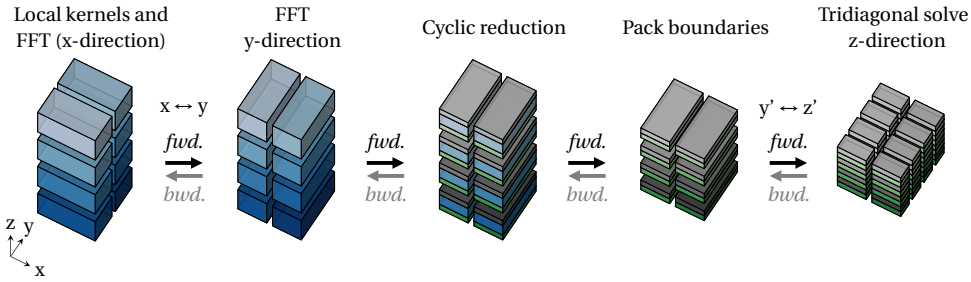


Figure 2.3: Modified parallel tridiagonal solver for large-scale DNS. The first two sub-images follow the same conventions as Figure 2.2, where the color of each partition indicates a different MPI task. In the third and fourth sub-images, the process of cyclic reduction is highlighted, by applying a different color to the boundary values for each partition.

While the approach presented in Figure 2.2 enables GPU-accelerated DNS of fluid flows on many CPUs/GPUs, the transposing operations may result in a major performance loss. This may be particularly problematic in modern GPU-based systems at scale, as the inter-node bandwidth is orders of magnitude slower than the GPU memory bandwidth or the intra-node communication. Hence, we need an approach that: (1) keeps a 2D parallelization, which is unavoidable at scale, and (2) reduces the amount of data used in collective communications to a reasonable minimum. We will explain this approach below.

SOLUTION WITH PARALLEL TRIDIAGONAL SOLVER

In this approach, we split the computational domain along the z direction, and exploit a parallel tridiagonal solver as in (K.-H. Kim et al., 2021, 2023; László et al., 2016) to circumvent the large *all-to-all* operations in the previous section. This approach is shown in Figure 2.3. This algorithm starts with a cyclic reduction step, such that only information

regarding the boundaries of every slice in the z-direction must be communicated to other MPI tasks. Subsequently, a tri-diagonal system for the boundary values is constructed. While an *all-to-all* type of collective is still needed, internal data points are not communicated, which can drastically reduce the communication overhead. The steps of this algorithm are summarized as follows:

1. Compute right-hand side term $d_{i,j,k}$ of eq. (2.12) in x-aligned pencils.
2. Perform $N_y N_z$ Fourier-based transforms along x (see Figure 2.3).
3. Transpose data to y-aligned pencils and perform $N_x N_z$ Fourier-based transforms along y.
4. Perform $N_x N_y$ cyclic reductions along z, pack $2N_x N_y$ boundary values, and transpose the packed boundaries to z-aligned pencils.
5. Solve $N_x N_y$ reduced systems of tridiagonal equations along z, with local size $2p_z$.
6. Transpose data to y-aligned pencils, unpack $2N_x N_y$ boundary values, and reconstruct the internal solution fields.
7. Perform the reciprocate inverse transforms and transpose operation as in step 3.
8. Perform the reciprocate inverse transforms as in step 2.

In the previous steps, $N_{x/y/z}$ is again the local grid size for each MPI rank in each Cartesian direction, whereas p_z is the number of divisions of the computational domain along the z-direction, which would correspond to $p_z = 4$ in Figure 2.3. The details of the parallel tridiagonal algorithm, and the modifications proposed in this work for the computation of its internal coefficients, are explained in Sections 2.2.2. Clearly, the amount of data transferred among MPI tasks is substantially reduced (K.-H. Kim et al., 2021). The tridiagonal system of equations found in the right-side of Figure 2.3 have a size of $2p_z$, where p_z is the number of partitions of the computational domain along the z-direction. Therefore, the parallel tridiagonal solver should be efficient as long as $2p_z \ll N_z$, where N_z is the total number of grid points in the z-direction.

Interestingly, in the parallel tridiagonal solver, increasing the number of lateral divisions (p_y) favors strong scalability: When p_y is increased, the size of the boundaries (per MPI task) is reduced as $2n_x n_y / p_y$. Therefore, doubling p_y halves the data communicated per task, leading to excellent scalability. In contrast, increasing the number of vertical partitions (p_z) does not reduce the MPI workload, and thus it is unfavorable for scalability. This is particularly relevant for 1D slab configurations where $p_y = 1$ and p_z is the total of GPUs. Still, 1D slab configurations are optimal when $p_z \ll N_z$.

PARALLEL TRIDIAGONAL ALGORITHM

Numerous approaches may be considered to parallelize the Thomas algorithm to solve a tridiagonal system (see, e.g., the survey in (K.-H. Kim et al., 2021)). Here we adopt the method proposed by (László et al., 2016), which uses cyclic reduction, combined with a Thomas algorithm for a reduced system. We have illustrated the approach in Figure 2.3, and summarize it below.

First, the distributed tridiagonal systems of this form (cf. eq. (2.12))

$$a_k \, \phi_{k-1} + b_k \, \phi_k + c_k \, \phi_{k+1} = d_k, \tag{2.13}$$

are locally reduced to a problem where inner unknowns within the computational subdomain are only a function of the values at its top and bottom boundary:

$$a'_k \, \phi_0 + \phi_k + c'_k \, \phi_{m-1} = d'_k, \tag{2.14}$$

using a cyclic reduction step. The original set of coefficients and right-hand-side (a, b, c, d) are then reduced to the (a', c', d'), where the main diagonal is normalized to have unit weight. Algorithm 1 describes this approach for completeness, and more details can be found in (K.-H. Kim et al., 2021, 2023; László et al., 2016).

Second, the sets of values (a', c', d') at the boundaries of every domain $k = \{0, m-1\}$ can be grouped and transposed in a collective operation (recall Figure 2.3). Then, the standard Thomas algorithm solves the reduced systems of tridiagonal equations for the boundary values of all subdomains. Finally, the boundary data for $\phi_{i,j,k}$ can be globally transposed, and the values of $\phi_{i,j,k}$ in the interior of every sub-domain can be reconstructed using eq. (2.14).

A few important notes should be considered to secure parallel performance at scale when combining the pressure Poisson equation with z-implicit diffusion. While the same computational approach may be taken for solving both cases, the straightforward implementation of Algorithm 1 would be far from optimal in both cases. Note that: (1) the reduced tridiagonal system for the Poisson equation is time-invariant (eq. (2.4)), with a problem that changes for each (i, j) index, yet (2) the z-implicit matrix is constant for each (i, j) index, but time-dependent (eq. (2.11)). Hence, a key optimization for the Poisson equation is to perform the transpose operations associated with the reduced system coefficients (a', b', c') only once as an initialization step (recall the penultimate step in Figure 2.2), as only d' varies with the right-hand-side of eq. (2.4). Regarding implicit z diffusion, (a', b', c') are time-dependent, but identical for all (i, j) indexes mapped to different GPU threads to solve the equations along z. Thus, rather than communicating (a', b', c') through MPI operations, it is much faster to have each GPU computing the global (a', b', c') coefficients corresponding to all MPI tasks aligned in the z-direction, and locally copy the values pertaining to its own subdomain. This is done on the GPUs with unnoticeable computational overhead, and effectively avoids expensive MPI communication operations.

Accordingly, to efficiently handle implicit z diffusion, Algorithm 1 was re-derived in a flavor that splits the solution into an initialization with pre-computed coefficients and a runtime step. This approach is presented in Algorithm 2. A major advantage of this approach is that only the array d' is modified in-place at runtime. This marks a large contrast to Algorithm 1, which requires thread-private arrays (or memory buffers) to track intermediate changes in the arrays (a', c'). This change is particularly relevant when solving for implicit 1D diffusion, since only 1D arrays with precomputed (a', b', c') coefficients can handle the solution process.

From a mathematical perspective, the new algorithm is derived by analyzing the cyclic reduction process, and carefully tracking which references to the (a', b', c') arrays can be

Algorithm 1 Cyclic reduction step of the parallel tridiagonal algorithm (K.-H. Kim et al., 2021; László et al., 2016), where **d** corresponds to the right-hand-side of the system (see Eq. (2.12)).

Step 1: Initialization
Input: $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$
$\mathbf{a}_0 \leftarrow \mathbf{a}_0/\mathbf{b}_0$; $\mathbf{c}_0 \leftarrow \mathbf{c}_0/\mathbf{b}_0$; $\mathbf{d}_0 \leftarrow \mathbf{d}_0/\mathbf{b}_0$
$\mathbf{a}_1 \leftarrow \mathbf{a}_1/\mathbf{b}_1$; $\mathbf{c}_1 \leftarrow \mathbf{c}_1/\mathbf{b}_1$; $\mathbf{d}_1 \leftarrow \mathbf{d}_1/\mathbf{b}_1$
for i=2,...,m-1 **do**
    $r \leftarrow 1/(\mathbf{b}_i - \mathbf{a}_i\mathbf{c}_{i-1})$
    $\mathbf{d}_i \leftarrow r(\mathbf{d}_i - \mathbf{a}_i\mathbf{d}_{i-1})$
    $\mathbf{c}_i \leftarrow r\mathbf{c}_i$
    $\mathbf{a}_i \leftarrow -r\mathbf{a}_i\mathbf{a}_{i-1}$
end for
for i=m-3,...,1 **do**
    $\mathbf{d}_i \leftarrow \mathbf{d}_i - \mathbf{c}_i\mathbf{d}_{i+1}$
    $\mathbf{a}_i \leftarrow \mathbf{a}_i - \mathbf{c}_i\mathbf{a}_{i+1}$
    $\mathbf{c}_i \leftarrow -\mathbf{c}_i\mathbf{c}_{i+1}$
end for
$r \leftarrow 1/(1 - \mathbf{a}_1\mathbf{c}_0)$
$\mathbf{d}_0 \leftarrow r(\mathbf{d}_0 - \mathbf{c}_0\mathbf{d}_1)$
$\mathbf{a}_0 \leftarrow r\mathbf{a}_0$
$\mathbf{c}_0 \leftarrow -r\mathbf{c}_0\mathbf{c}_1$
$\mathbf{b} = \mathbf{1}$

Step 2: Solve reduced system of equations for boundary values

Step 3: Reconstruct the solution in-place
Input: $\mathbf{a}, \mathbf{c}, \mathbf{d}, \mathbf{x}_0, \mathbf{x}_{m-1}$
$\mathbf{d}_0 \leftarrow \mathbf{x}_0$
$\mathbf{d}_{m-1} \leftarrow \mathbf{x}_{m-1}$
for i=1,...,m-2 **do**
    $\mathbf{d}_i \leftarrow \mathbf{d}_i - \mathbf{a}_i\mathbf{x}_0 - \mathbf{c}_i\mathbf{x}_{m-1}$
end for

replaced by either their input or output values. After making this distinction, it becomes evident that the reduction process for the array d' does not depend on intermediate values for (a', b', c') being over-written. Therefore, it is natural to split the process into initialization and runtime stages. Moreover, it is important to highlight that only the initial values of (a', c') are used. In the DNS solver, the variables (a', c') always correspond to 1D vectors, even for Poisson or Helmholtz solvers. As a result, storing the initial values of (a, c) creates a negligible performance overhead.

Algorithm 2 Alternative flavor of the cyclic reduction method proposed in this work, with a separation between initialization and runtime operations. The variables ($\mathbf{A}, \mathbf{B}, \mathbf{C}$) correspond to the original coefficients of the tridiagonal equations, whereas ($\mathbf{a}, \mathbf{b}, \mathbf{c}$) are the modified coefficients after cyclic reduction.

Step 1: Initialization
Input: $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{a}, \mathbf{b}, \mathbf{c}$
$\mathbf{a}_1 \leftarrow A_1$
$\mathbf{b}_1 \leftarrow \mathbf{B}_1$
for i=2,...,m-1 **do**
    $\mathbf{b}_i \leftarrow \mathbf{B}_i - A_i \; C_{i-1} / \mathbf{b}_{i-1}$
    $\mathbf{a}_i \leftarrow -A_i \; \mathbf{a}_{i-1} / \mathbf{b}_{i-1}$
end for
$\mathbf{c}_{m-1} \leftarrow C_{m-1}$
$\mathbf{c}_{m-2} \leftarrow C_{m-2}$
for i=m-3,...,1 **do**
    $\mathbf{a}_i \leftarrow \mathbf{a}_i - C_i \; \mathbf{a}_{i+1} / \mathbf{b}_{i+1}$
    $\mathbf{c}_i \leftarrow -C_i \; \mathbf{c}_{i+1} / \mathbf{b}_{i+1}$
end for
$\mathbf{a}_0 \leftarrow A_0$
$\mathbf{b}_0 \leftarrow \mathbf{B}_0 - C_0 \; \mathbf{a}_1 / \mathbf{B}_1$
$\mathbf{c}_0 \leftarrow -C_0 \; \mathbf{c}_1 / \mathbf{B}_1$
▷ Note: To avoid GPU divisions in Steps 2 and 4, optimized implementations can store $1/\mathbf{b}$ instead of $\mathbf{b}$.

Step 2: Runtime reduction
Input: $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$
for i=2,...,m-1 **do**
    $\mathbf{d}_i \leftarrow \mathbf{d}_i - A_i \; \mathbf{d}_{i-1} / \mathbf{b}_{i-1}$
end for
for i=m-3,...,0 **do**
    $\mathbf{d}_i \leftarrow \mathbf{d}_i - C_i \; \mathbf{d}_{i+1} / \mathbf{b}_{i+1}$
end for

Step 3: Solve reduced system of equations for boundary values

Step 4: Reconstruct the solution in-place
Input: $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{x}_0, \mathbf{x}_{m-1}$
$\mathbf{d}_0 \leftarrow \mathbf{x}_0$
$\mathbf{d}_{m-1} \leftarrow \mathbf{x}_{m-1}$
for i=1,...,m-2 **do**
    $\mathbf{d}_i \leftarrow (\mathbf{d}_i - \mathbf{a}_i \mathbf{x}_0 - \mathbf{c}_i \mathbf{x}_{m-1}) / \mathbf{b}_i$
end for

2.2.3. IMPLEMENTATION

As previously noted, the approaches we have presented are implemented in the open-source code CaNS (Costa, 2018; Costa et al., 2021). CaNS is written in Modern Fortran, and since its version *2.0* offloads data and computation to GPUs using OpenACC. At the fine-grained parallelization level of the TDMA implementations, each i, j index is assigned to a thread, which serially performs operations along the third domain direction. The CPU implementation uses the *2DECOMP&FFT* library (Rolfo et al., 2023) to perform transposes. Conversely, the multi-GPU implementation uses the *cuDecomp* library for pencil-distributed calculations at scale that feature transposes and halo exchanges (Romero et al., 2022) is used in CaNS. The main advantage of *cuDecomp* is its runtime autotuning capabilities, which allows to confidently select a well-performing combination of 2D processor grid and communication backend (with several low-level implementations of the transposing algorithm in CUDA-aware MPI, NCCL, or NVSHMEM). *cuDecomp*'s flexibility allowed for a very straightforward implementation of the communication operations that can be visualized in Figure 2.2. Indeed, the *all-to-all* operations needed to communicate the boundary values for each sub-group of slices along the z-direction could be replaced by the existing transpose operations available in the *cuDecomp* or *2DECOMP* libraries.

The distributed-memory implementation of CaNS, using *cuDecomp* and *cuFFT*, allowed for very efficient calculations at scale on NVIDIA-based systems. However, in the present work we decided to benchmark our approach on the supercomputers Leonardo (NVIDIA-based) and LUMI (AMD-based). We summarize our implementation approach for the latter, which we plan to incorporate in the CaNS public repository in the near future.

Regarding the verification of the implementation, CaNS has been extensively validated in the past (Costa, 2018; Costa et al., 2021). Therefore, the correctness of the current implementation can be trivially verified, since only the parallel tridiagonal solvers have been touched. An explicit comparison with respect to the output of the full-transpose method is accurate up to machine precision for the modified subroutines, which verifies the correctness of the implementation.

MANY-GPU IMPLEMENTATION ON LUMI

On LUMI, the Cray Fortran compiler is readily compatible with OpenACC, and GPU-aware MPI is supported to perform data transfer among GPU devices (or GCDs). However, since LUMI has AMD cards, some work was required to port the transpose and halo exchange algorithms for LUMI. One approach would be to adjust *cuDecomp* such that the NVIDIA-specific features of the library are masked out of the build workflow. In the present work, we took a different route and developed a cross-platform communication library based on Fortran and OpenACC named *diezDecomp*. Its source code is available on GitHub under an MIT license (Diez, 2025). This implementation uses OpenACC and GPU-aware MPI to perform transpose and halo exchange operations. During transpose operations, *diezDecomp* has been optimized to pack and unpack data related to different MPI ranks simultaneously (in parallel GPU threads), and it supports conversions between different indexing orders (e.g., $x/y/z$ to $y/x/z$). As further verification, the performance of the *diezDecomp* library was tested in the Leonardo supercomputer, achieving nearly identical running times as the *cuDecomp* library.

Finally, it is important to highlight that the current distributed Poisson/Helmholtz solver is able to work with various types of FFT libraries, such as *cuFFT* ("cuFFT API Reference", 2024), *hipFFT* ("hipFFT documentation", 2024), or even in-house FFT implementations. In this context, our porting effort uses the *hipFFT* library for simulations in AMD GPUs using the bindings provided by the *hipFort* project (**hipFort**), whereas *cuFFT* is enabled for NVIDIA GPUs via the CUDA toolkit ("CUDA Fortran for Scientists and Engineers", 2014). Since both FFT libraries have similar APIs, our implementation uses CPP macros to switch between libraries depending on the target platform.

2.3. RESULTS

2.3.1. STRONG AND WEAK SCALABILITY

We consider a large-scale turbulent plane channel flow setup, which exercises all important steps presented in this chapter, including z-implicit diffusion. Figure 2.4 presents a strong scalability test for a $N_x \times N_y \times N_z = 7168 \times 7168 \times 1594$ grid, containing approximately 80 billion grids points. This corresponds to a channel with a domain size of $L_x \times L_y \times L_z = 12.8 \times 6.4 \times 2$, with a friction Reynolds number $Re_\tau \approx 5\,000$ (Lee & Moser, 2015). Note that, in many GPU-resident workloads, a strong scaling test likely shows performance deterioration, as the occupancy of each GPU is being lowered (Romero et al., 2022). The tests were performed on the Leonardo and LUMI supercomputers, which again, feature NVIDIA A100 and AMD MI250X GPU cards, respectively. In the figure, we compare the full transpose method (FTM) to the approach with the parallelized tridiagonal matrix algorithm (P-TDMA), for both 1D (slab) decompositions (as in Figure 2.2 without decomposing along x, and in Figure 2.3 without decomposing along y) and for 2D pencil decomposition schemes with different levels of process decomposition along z: $p_z = 2$ and 128. The number of partitions along the y-direction (p_y) are set from the total number of GPUs: $n_g = p_y \times p_z$. These configurations are intended to show how the solver scales in the two limits where the 2D pencil decomposition has either the least number of partitions along z, or very high values. Clearly, the P-TDMA algorithm is far more efficient than FTM when 2D pencil decompositions are considered. This is explained by Figures 2.2 and 2.3, because the P-TDMA algorithm transposes a significantly smaller amount of data than the FTM method. Focusing on the performance of the Poisson solver (bottom panels of Figure 2.4), there is a consistent 1.5× improvement in wall-clock time per step. While strong scaling deterioration is expected, it is interesting to note that some of the curves show reasonably mild deviations from the ideal scaling curve.

The P-TDMA method shows great efficiency for $p_z = 2$, and almost matches the best-performing 1D slab configurations in the FTM case. This is expected, as the P-TDMA method only transfers data along one small boundary in the z-direction when $p_z = 2$, with a computational cost roughly similar to a halo exchange.

Let us now consider the difference between the wall-clock time of the Poisson solver and the full time steps. The most important difference here is the solution of three separate systems of 1D implicit diffusion equations in the wall-clock time of the full time step. Therefore, configurations that transfer more data when solving for 1D implicit diffusion tend to have worse performance, such as the FTM approach with a 2D pencil decomposition.

The wall-time per steps for the Poisson solver with 1 024 GPUs/GCDs converge to a similar value for both the best FTM configurations and the P-TDMA cases with 2D pencil decompositions, since their performance is dominated by the transpose operations in the x-y directions. Interestingly, one can notice a major performance gain for the P-TDMA when $p_z = 128$ on LUMI, which corresponds to a special case where 50% of the required data after the x-y transpose is already present in the GPU, requiring only a local copy.

In the case of the P-TDMA algorithm with 1D slabs, it can be observed in Figure 2.4 that the running times remain nearly constant when the number of GPUs/GCDs increases from 256 to 512. This is due to the P-TDMA method having boundaries with a fixed size of $(N_x \times N_z \times 2)$, and thus, the data transferred per GPU/GCD remains equal as the number of processes is increased. Despite this, the P-TDMA method with 1D slabs still outperforms the FTM approach, when the number of grid points along the z-direction is very large for every 1D slab. This is illustrated in Figure 2.5, where a strong scalability benchmark is presented for 1D slabs for $Re_\tau \approx 2500$ and a grid size of $N_x \times N_y \times N_z = 3200 \times 3456 \times 900$. In this benchmark, the P-TDMA method outperforms the FTM approach with 32 GPUs/GCDs, yet its performance decreases as the number of processes grows. For cases at lower Reynolds numbers, the performance improvements can be expected to grow in favor of the P-TDMA method, since the number of grid points per GPU along z increases for the 1D slabs.

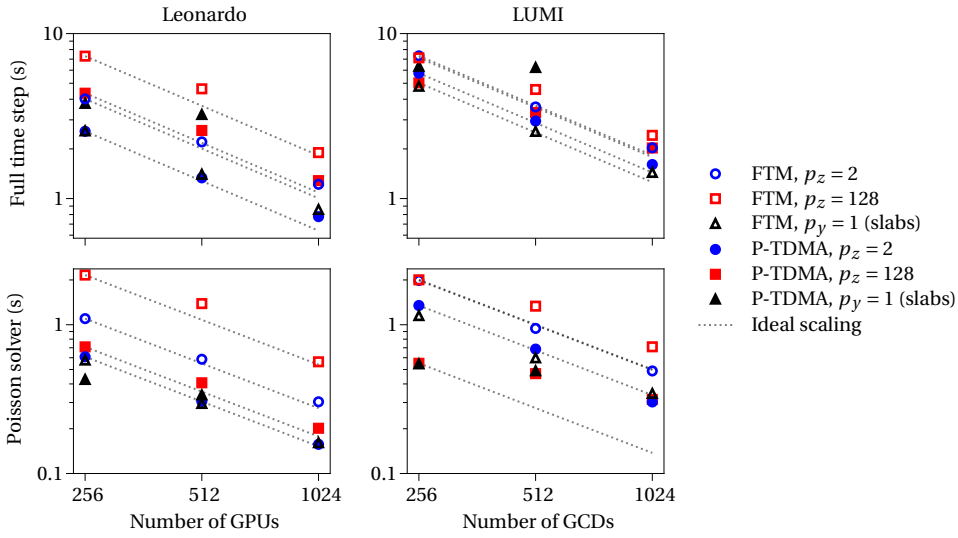


Figure 2.4: Strong scalability chart for a wall-bounded flow with grid size $(N_x \times N_y \times N_z) = (7168 \times 7168 \times 1594)$, which roughly corresponds to a friction Reynolds number of $Re_\tau \approx 5000$. The variable p_z corresponds to the number of divisions along the z direction for the pencil decomposition scheme. The abbreviation "FTM" refers to the original DNS solver, which was based on the full-transpose method. All simulations were performed in the Leonardo and LUMI supercomputers. Please note that P-TDMA method with 1D slabs (filled black triangles) cannot be run with 1 024 GPUs/GCDs, due to an insufficient number of grid points in the z-direction: $N_z/p_z < 2$.

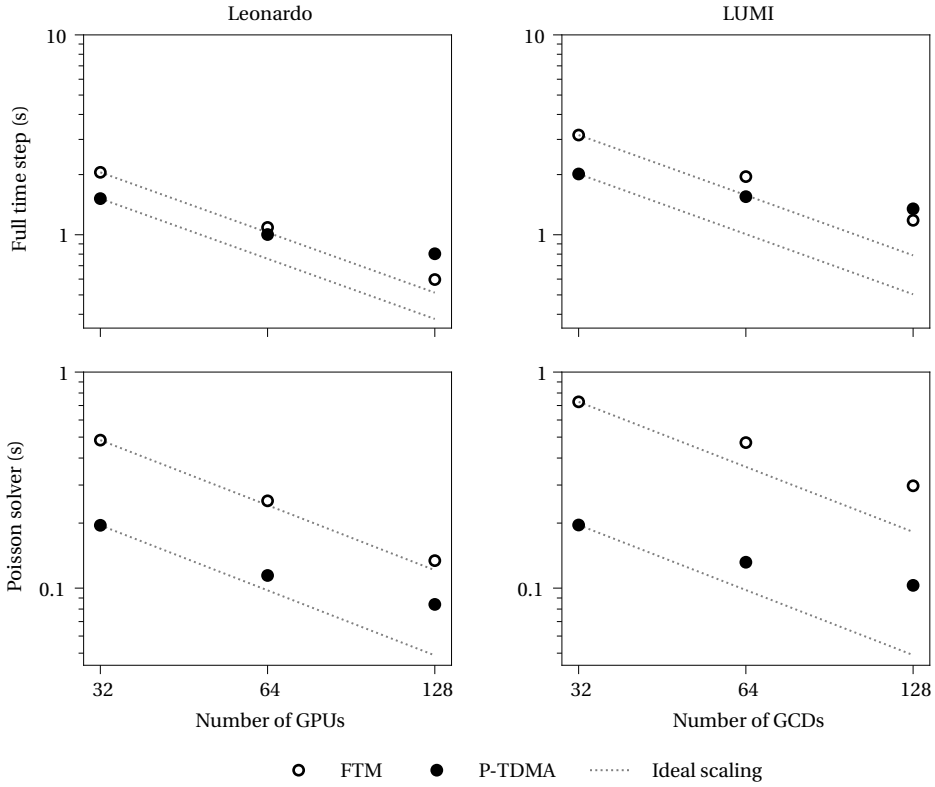


Figure 2.5: Strong scalability chart for a wall-bounded flow with grid size $(N_x \times N_y \times N_z) = (3200 \times 3456 \times 900)$, which roughly corresponds to a friction Reynolds number of $Re_\tau \approx 2{,}500$. All runs correspond to 1D slab configurations, as in Figure 2.2 without decomposing along x, and in Figure 2.3 without decomposing along y.

The results from the weak scaling tests are shown in Figure 2.6. Weak scaling is the most important scaling indicator in large-scale GPU-resident DNS, as optimal resource usage requires maximizing GPU occupancy, which is kept fixed in these tests. We fixed the local domain sizes to about 318.5 million grid points per GPU, by considering a grid with $3456 \times 3072 \times 30$ points per GPU, which saturates the GPUs (GCDs) on Leonardo (LUMI). For the P-TDMA method, the number of partitions p_z along the z-direction is equal to the number of GPUs/GCDs, since this corresponds to the most challenging scenario for the weak scaling analysis; for FTM, we use the optimal configuration for 1D slabs. Clearly, the performance of the P-TDMA approach is superior, not only being approximately 2× faster in wall-clock time per step, but also in terms of weak scaling. An 8-fold increase in the GPUs/GCDs used results in a 3% (13%) performance degradation on Leonardo (LUMI), while the full transpose method shows a major weak scaling loss of 35% (52%) on Leonardo (LUMI). This shows that, indeed, the proposed improvements are important for efficient wall turbulence simulations at scale.

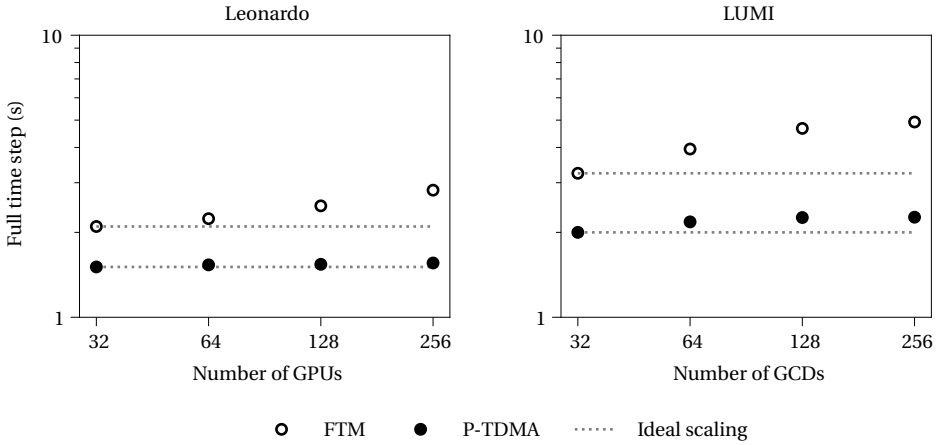


Figure 2.6: Weak scalability analysis for DNS cases with a fixed grid size $(N_x \times N_y \times N_z) = (3456 \times 3072 \times 30)$ per GPU device. For the P-TDMA method, the pencil decomposition scheme only considers one partition along the y-direction ($p_y = 1$), and a number of z-partitions (p_z) equal to the number of GPUs. The FTM approach considers the 1D slab configuration as in Figure 2.2 without decomposing along x.

2.3.2. BREAKDOWN OF PARALLEL PERFORMANCE

Let us now understand in more detail the results presented in the previous sections. Figure 2.7, compares the workload distribution of the current approach and the previous algorithm based on the full-transpose method. The DNS cases chosen for comparison are simulations with 1 024 GPUs/GCDs, shown in the strong scalability chart (Figure 2.4). Consistently with the previous observations, the performance differences in the supercomputers we have tested show similar trends overall. The 2D pencil decomposition considered for comparison is $(p_y \times p_z) = (512, 2)$, which has the lowest running times overall for the P-TDMA algorithm. Not surprisingly, the P-TDMA approach is much faster than the full-transpose algorithm while solving for the pressure-Poisson equation. However, the P-TDMA algorithm is slower when solving for 1D implicit diffusion alongside each velocity component (u, v, w). This is also expected, as again, the FTM uses an initial z-aligned decomposition, where the full tridiagonal problems are local to each MPI task. Both algorithms show similar overhead associated with halo exchanges, which is also expected.

AVERAGE TIMINGS

When inspecting the contributions to the mean total wall-time per step in Figure 2.7, it becomes clear that *all-to-all* operations are the main performance bottleneck of the full-transpose method, whereas the P-TDMA approach is far less communication-intensive. Interestingly, the Poisson solver with the P-TDMA algorithm spends 81.7% (89.6%) of the time on Leonardo (LUMI) performing local transposes along the x-y plane, which are required between the Fourier-based transforms described in Section 2.2.2. The *all-to-all* operations associated with the parallel tridiagonal solver only require 0.2% (0.6%) of the computing time in Leonardo (LUMI), as this collective operation only communicates

boundary values for each MPI task (recall the penultimate step in Figure 2.2). Note that, while the time spent performing $x \rightarrow y$ transposes can be (partially) reduced by minimizing the p_y divisions along the y-direction (e.g., $p_y = 2$), the overall performance degrades in Figure 2.4 when the number of vertical partitions p_z is increased for this case. This trend is valid for both the FTM approach and the P-TDMA methods, since the cost of performing y-z transposes increases as p_z grows. Therefore, a careful trade-off must be considered. Moreover, the optimal DNS domain decomposition is not only problem-dependent but also hardware-dependent, which makes runtime tuning of the computational setup very relevant (Romero et al., 2022).

Finally, in Figure 2.7, it can be noticed that the P-TDMA approach also performs $x \rightarrow y$ transposes when solving for implicit 1D diffusion, with a small but noticeable communication footprint. This is not strictly necessary, since cyclic reduction can be directly performed using the initial x-aligned pencils (Figure 2.3, left), and then a direct $x \rightarrow z$ transpose could be used to obtain z-aligned pencils (Zhu et al., 2018). This direct transpose is less trivial to implement and is not featured in *cuDecomp* or *2DECOMP&FFT*. Yet, since the *diezDecomp* communication library supports $x \rightarrow z$ transposes for any desired 2D decomposition, we tested the performance gains of this direct transpose. The results of this additional benchmark are shown in A.2 for LUMI. While the computational overhead of the additional $x \rightarrow y$ transpose is small for the cases shown in Figure 2.7 with the decomposition $(p_y \times p_z) = (512 \times 2)$, we identified that other pencil decompositions suffered from higher performance losses. For instance, the benchmark shows that the DNS case with $(p_y \times p_z) = (8 \times 128)$ is about 55% faster during the calculation of z-implicit diffusion when using direct $x \rightarrow z$ transposes. This results in savings of about 18% in the total wall-clock time per step.

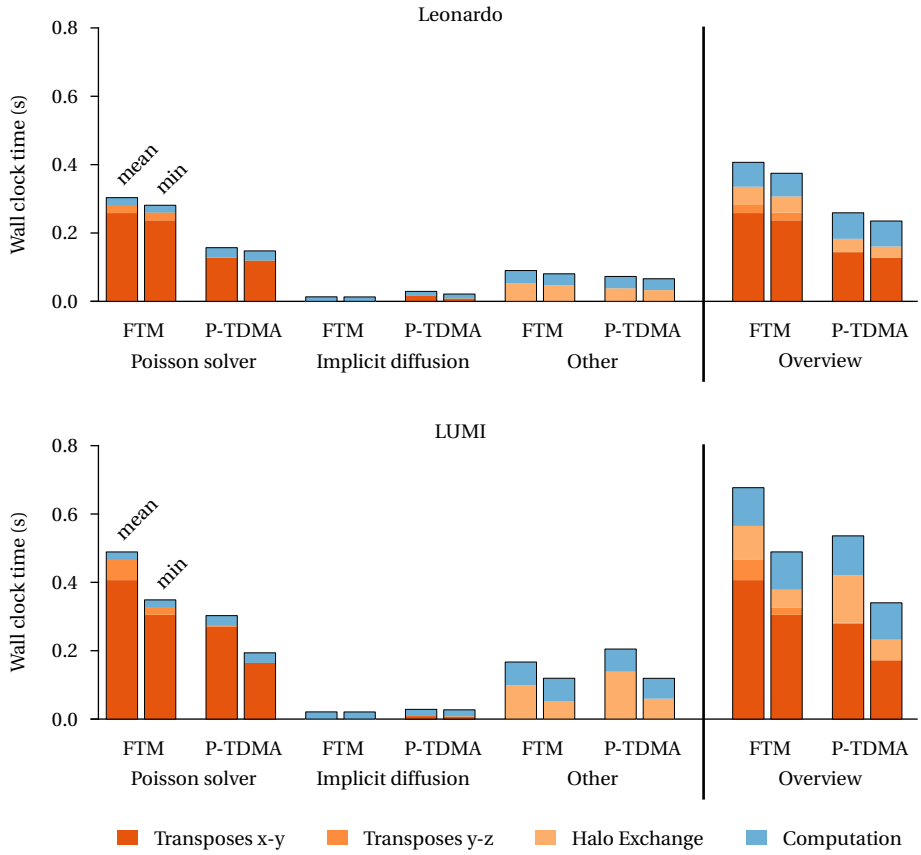


Figure 2.7: Comparison between the GPU timer profiles for the DNS solver with the P-TDMA and FTM methods on Leonardo and LUMI. For each profiled operation, the left bars correspond to the average wall-time per step (*mean*), while the right bars are the aggregated results considering the best elapsed times per operation (*min*). The data was extracted from the DNS runs with 1 024 GPUs (Leonardo) or GCDs (LUMI), which are presented in the strong scalability chart (Figure 2.4) for a fixed grid size $(N_x \times N_y \times N_z) = (7168 \times 7168 \times 1594)$. For the DNS solver with P-TDMA, the data further corresponds to the case with $(p_y \times p_z) = (512 \times 2)$ partitions along the y and z directions, which has the fastest running times in the strong scalability chart. Both GPU timer profiles correspond to the averaged results for a single RK3 sub-step within the DNS solvers. Please note that the $y \leftrightarrow z$ transposes for the P-TDMA algorithm transfer much less data, and thus their impact on the plotted budgets is minor.

BEST ELAPSED TIMES PER OPERATION

To better explain the differences between systems, we included in Figure 2.7 the aggregated best elapsed times per operation (right bars). Remarkably, the total running times for the FTM (P-TDMA) method are decreased by about 28% (37%) for LUMI, and by 8% (9%) for Leonardo. It appears that the average wall-clock time per step on LUMI is significantly suboptimal compared to that on Leonardo.

Significant performance differences between these machines for *all-to-all* collective operations have been reported in (De Sensi et al., 2024), where Leonardo outperformed LUMI by almost 50% in goodput benchmarks at scale. We argue that the analysis considering the best elapsed times per operation resembles a goodput benchmark better, since these results contain the smallest levels of network delays possible. Interestingly, when inspecting the aggregated best elapsed times (*min*) in Figure 2.7, it can be noticed that LUMI is only 31% (45%) slower than Leonardo for the total running times of the FTM (P-TDMA) method, consistently with the trends in (De Sensi et al., 2024). We verified that this trend is robust across other cases (not shown).

2.4. CONCLUSIONS

We have presented a numerical approach for GPU-based massively-parallel DNS of turbulent wall flows with one inhomogeneous direction. Using the CaNS solver as base, we extended it with a parallel tridiagonal algorithm for solving the pressure Poisson equation, and to handle implicit integration of the wall-normal diffusion term. To achieve this, we adopted a recently-proposed approach for solving distributed tridiagonal systems (K.-H. Kim et al., 2021; László et al., 2016), and implemented it in a pencil-decomposed framework. Allowing for two-dimensional decompositions is key, as slab-decomposed approaches are bound to breakdown for DNS at sufficiently high Reynolds number.

Carefully handling z-implicit diffusion was key to secure the improved parallel performance at scale. To this end, we have proposed a re-worked flavor of the original parallel cyclic reduction – TDMA approach presented in (László et al., 2016). We have shown that, by re-working the algorithm into a pre-processing and runtime step, one can easily solve the three linear systems per time iteration associated with this implicit discretization.

We have tested the different approaches at scale, using up to 1 024 GPUs/GCDs on the supercomputers Leonardo and LUMI. The results of the scalability tests reveal that the new distributed Poisson solver shows compelling performance gains for 2D pencil decompositions, being approximately twice faster in the LUMI and Leonardo supercomputers than the original CaNS version based on the full-transpose approach using 1 024 GPUs/GCDs. A detailed analysis of the GPU timer profiles reveals that the performance improvements are largely due to the reduced size of the global all-to-all transpose operations among MPI tasks. The scalability of the DNS solver was also tested in large-scale simulations of wall-bounded flows, benchmarking the performance of entire physical time steps. At scale, the new approach was found to be approximately 1.5× faster in the LUMI and Leonardo supercomputers with a 2D pencil decomposition of $(p_y \times p_z) = (512 \times 2)$ while completing entire physical time steps. The observed performance differences between the two machines were understood by inspecting the best recorded times per algorithm step, showing that LUMI runs experienced higher latency than Leonardo. In general, we find that minimizing the number of p_z partitions in 2D pencil decompositions reduces the

running times for the DNS solver with either the full-transpose method or the parallel tridiagonal algorithm. This is attributed to the reduced cost of performing transposes in the y-z directions. Additionally, we highlight that the DNS solver, coupled with the parallel tridiagonal algorithm, can be configured to work with 2D pencil decompositions achieving identical running times as the most optimized 1D slab configurations available for medium-scale systems.

Regarding implementation, while the underlying numerical solver works on Leonardo out-of-the-box, some work was required to successfully run it on LUMI. As a by-product of the present effort, a cross-platform communication library *diezDecomp* was developed for halo exchanges and any-to-any transpose operations between MPI ranks with mismatched local problem sizes. While we could have achieved the same by modifying *cuDecomp*, we found some advantages in having a simpler library in Modern Fortran as an alternative with fewer dependencies.

Overall, this approach will enable DNS of turbulent wall flows at unprecedented scales, helping to bridge the gap between current setups that can be studied using first-principles simulations, and important applications in environmental and engineering systems.

2.5. ACKNOWLEDGEMENTS

The performance tests on the Leonardo supercomputer, based at CINECA (Italy), were enabled by the EuroHPC grant no. `EHPC-EXT-2022E01-054`. Access to the LUMI supercomputer was granted through the project no. `EHPC-DEV-2024D04-039`, and the NWO Large Compute Grant no. `EINF-10608`. We also acknowledge NWO for providing access to the Snellius supercomputer, based at SURF (Netherlands). We thank Josh Romero, Massimiliano Fatica, and Sergio Pirozzoli for the insightful discussions. Finally, we would also like to thank the two anonymous Reviewers for their careful review, which contributed to the improvement of this chapter.

REFERENCES

AMD Instinct MI250X Accelerators. (2025). AMD. https://www.amd.com/en/products/accelerators/instinct/mi200/mi250x.html

Bernardini, M., Modesti, D., Salvadore, F., & Pirozzoli, S. (2021). Streams: A high-fidelity accelerated solver for direct numerical simulation of compressible turbulent flows. *Computer Physics Communications*, *263*, 107906. https://doi.org/https://doi.org/10.1016/j.cpc.2021.107906

Breugem, W. P., & Boersma, B. J. (2005). Direct numerical simulations of turbulent flow over a permeable wall using a direct and a continuum approach. *Physics of Fluids*, *17*(2), 025103. https://doi.org/10.1063/1.1835771

Chorin, A. J. (1967). A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, *2*(1), 12–26. https://doi.org/https://doi.org/10.1016/0021-9991(67)90037-X

Costa, P. (2018). A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows. *Computers & Mathematics with Applications*, *76*(8), 1853–1862. https://doi.org/https://doi.org/10.1016/j.camwa.2018.07.034

Costa, P., Phillips, E., Brandt, L., & Fatica, M. (2021). Gpu acceleration of cans for massively-parallel direct numerical simulations of canonical fluid flows [Development and Application of Open-source Software for Problems with Numerical PDEs]. *Computers & Mathematics with Applications*, *81*, 502–511. https://doi.org/https://doi.org/10.1016/j.camwa.2020.01.002

Cuda fortran for scientists and engineers. (2014). In M. Fatica & G. Ruetsch (Eds.), *Cuda fortran for scientists and engineers*. Morgan Kaufmann.

cuFFT API reference. (2024). https://docs.nvidia.com/cuda/cufft

De Sensi, D., Pichetti, L., Vella, F., De Matteis, T., Ren, Z., Fusco, L., Turisini, M., Cesarini, D., Lust, K., Trivedi, A., Roweth, D., Spiga, F., Di Girolamo, S., & Hoefler, T. (2024). Exploring gpu-to-gpu communication: Insights into supercomputer interconnects. *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–15. https://doi.org/10.1109/SC41406.2024.00039

De Vanna, F., Avanzi, F., Cogo, M., Sandrin, S., Bettencourt, M., Picano, F., & Benini, E. (2023). Uranos: A gpu accelerated navier-stokes solver for compressible wall-bounded flows. *Computer Physics Communications*, *287*, 108717. https://doi.org/https://doi.org/10.1016/j.cpc.2023.108717

Diez, R. (2025). diezDecomp: Cross-platform library for transposes and halo exchanges in extreme-scale simulations. https://github.com/Rafael10Diez/diezDecomp

Diez Sanhueza, R., Peeters, J., & Costa, P. (2025). A pencil-distributed finite-difference solver for extreme-scale calculations of turbulent wall flows at high reynolds number. *Computer Physics Communications*, *316*, 109811. https://doi.org/https://doi.org/10.1016/j.cpc.2025.109811

Fadlun, E., Verzicco, R., Orlandi, P., & Mohd-Yusof, J. (2000). Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, *161*(1), 35–60. https://doi.org/https://doi.org/10.1006/jcph.2000.6484

Gong, Z., Deng, G., An, C., Wu, Z., & Fu, X. (2022). A high order finite difference solver for simulations of turbidity currents with high parallel efficiency. *Computers & Mathematics with Applications*, *128*, 21–33.

hipFFT documentation. (2024). https://rocm.docs.amd.com/projects/hipFFT/en/latest

Ishihara, T., Gotoh, T., & Kaneda, Y. (2009). Study of high–reynolds number isotropic turbulence by direct numerical simulation. *Annual Review of Fluid Mechanics*, *41*(1), 165–180. https://doi.org/10.1146/annurev.fluid.010908.165203

Kim, J., & Moin, P. (1985). Application of a fractional-step method to incompressible navier-stokes equations. *Journal of Computational Physics*, *59*(2), 308–323. https://doi.org/https://doi.org/10.1016/0021-9991(85)90148-2

Kim, K.-H., Kang, J.-H., Pan, X., & Choi, J.-I. (2021). Pascal_tdma: A library of parallel and scalable solvers for massive tridiagonal systems. *Computer Physics Communications*, *260*, 107722. https://doi.org/https://doi.org/10.1016/j.cpc.2020.107722

Kim, K.-H., Kang, J.-H., Pan, X., & Choi, J.-I. (2023). Pascal_tcs: A versatile solver for large-scale turbulent convective heat transfer problems with temperature-dependent fluid properties. *Computer Physics Communications*, *290*, 108779. https://doi.org/https://doi.org/10.1016/j.cpc.2023.108779

Kim, Y., Ghosh, D., Constantinescu, E. M., & Balakrishnan, R. (2023). Gpu-accelerated DNS of compressible turbulent flows. *Computers & Fluids*, *251*, 105744. https://doi.org/https://doi.org/10.1016/j.compfluid.2022.105744

László, E., Giles, M., & Appleyard, J. (2016). Manycore algorithms for batch scalar and block tridiagonal solvers. *ACM Trans. Math. Softw.*, *42*(4). https://doi.org/10.1145/2830568

Lee, M., & Moser, R. D. (2015). Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *Journal of Fluid Mechanics*, *774*, 395–415. https://doi.org/10.1017/jfm.2015.268

Moin, P., & Verzicco, R. (2016). On the suitability of second-order accurate discretizations for turbulent flow simulations [Vortical Structures and Wall Turbulence]. *European Journal of Mechanics - B/Fluids*, *55*, 242–245. https://doi.org/https://doi.org/10.1016/j.euromechflu.2015.10.006

Nickolls, J., & Dally, W. J. (2010). The gpu computing era. *IEEE Micro*, *30*(2), 56–69. https://doi.org/10.1109/MM.2010.41

Nvidia A100 tensor core gpu: Unprecedented acceleration at every scale. (2024). Nvidia. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf

Orszag, S. A., & Patterson, G. S. (1972). Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett.*, *28*, 76–79. https://doi.org/10.1103/PhysRevLett.28.76

Pirozzoli, S., & Orlandi, P. (2021). Natural grid stretching for DNS of wall-bounded flows. *Journal of Computational Physics*, *439*, 110408. https://doi.org/https://doi.org/10.1016/j.jcp.2021.110408

Pope, S. B. (2001). Turbulent flows. *Measurement Science and Technology*, *12*(11), 2020–2021.

Reed, D., Gannon, D., & Dongarra, J. (2023). Hpc forecast: Cloudy and uncertain. *Commun. ACM*, *66*(2), 82–90. https://doi.org/10.1145/3552309

Rolfo, S., Flageul, C., Bartholomew, P., Spiga, F., & Laizet, S. (2023). The 2DECOMP&FFT library: an update with new CPU/GPU capabilities. *Journal of Open Source Software*, *8*(91), 5813. https://doi.org/10.21105/joss.05813

Romero, J., Costa, P., & Fatica, M. (2022). Distributed-memory simulations of turbulent flows on modern gpu systems using an adaptive pencil decomposition library. *Proceedings of the Platform for Advanced Scientific Computing Conference*. https://doi.org/10.1145/3539781.3539797

Salvadore, F., Bernardini, M., & Botti, M. (2013). Gpu accelerated flow solver for direct numerical simulation of turbulent flows. *Journal of Computational Physics*, *235*, 129–142. https://doi.org/https://doi.org/10.1016/j.jcp.2012.10.012

Schieffer, G., Shi, R., Markidis, S., Herten, A., Faj, J., & Peng, I. (2025). Understanding data movement in amd multi-gpu systems with infinity fabric. *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, 567–576. https://doi.org/10.1109/SCW63240.2024.00079

Schumann, U., & Sweet, R. A. (1988). Fast fourier transforms for direct solution of poisson's equation with staggered boundary conditions. *Journal of Computational Physics*, *75*(1), 123–137. https://doi.org/https://doi.org/10.1016/0021-9991(88)90102-7

TOP500 list - november 2024. (2024, November). https://www.top500.org/lists/top500/list/2024/11

Tryggvason, G., Scardovelli, R., & Zaleski, S. (2011). *Direct numerical simulations of gas–liquid multiphase flows*. Cambridge university press.

Uhlmann, M. (2005). An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, *209*(2), 448–476. https://doi.org/https://doi.org/10.1016/j.jcp.2005.03.017

Wray, A. A. (1986). *Very low storage time-advancement schemes* (tech. rep.). Internal Report, Moffett Field, CA, NASA-Ames Research Center.

Yang, M., Kang, J.-H., Kim, K.-H., Kwon, O.-K., & Choi, J.-I. (2023). Pascal_tdma 2.0: A multi-gpu-based library for solving massive tridiagonal systems. *Computer Physics Communications*, *290*, 108785. https://doi.org/https://doi.org/10.1016/j.cpc.2023.108785

Yang, M., Oh, G., Xu, T., Kim, J., Kang, J.-H., & Choi, J.-I. (2023). Multi-gpu-based real-time large-eddy simulations for urban microclimate. *Building and Environment*, *245*, 110856. https://doi.org/https://doi.org/10.1016/j.buildenv.2023.110856

Yeung, P. K., & Ravikumar, K. (2020). Advancing understanding of turbulence through extreme-scale computation: Intermittency and simulations at large problem sizes. *Phys. Rev. Fluids*, *5*, 110517. https://doi.org/10.1103/PhysRevFluids.5.110517

Zhu, X., Phillips, E., Spandan, V., Donners, J., Ruetsch, G., Romero, J., Ostilla-Mónico, R., Yang, Y., Lohse, D., Verzicco, R., Fatica, M., & Stevens, R. J. (2018). Afid-gpu: A versatile navier–stokes solver for wall-bounded turbulent flows on gpu clusters. *Computer Physics Communications*, *229*, 199–210. https://doi.org/https://doi.org/10.1016/j.cpc.2018.03.026

3

DATA-DRIVEN RANS TURBULENCE MODELS FOR VARIABLE PROPERTY FLOWS

This chapter presents a machine learning methodology to improve the predictions of traditional RANS turbulence models in channel flows subject to strong variations in their thermophysical properties. The developed formulation contains several improvements over the existing Field Inversion Machine Learning (FIML) frameworks described in the literature. We first showcase the use of efficient optimization routines to automatize the process of field inversion in the context of CFD, combined with the use of symbolic algebra solvers to generate sparse-efficient algebraic formulas to comply with the discrete adjoint method. The proposed neural network architecture is characterized by the use of an initial layer of logarithmic neurons followed by hyperbolic tangent neurons, which proves numerically stable. The machine learning predictions are then corrected using a novel weighted relaxation factor methodology, that recovers valuable information from otherwise spurious predictions. Additionally, we introduce L2 regularization to mitigate over-fitting and to reduce the importance of non-essential features. In order to analyze the results of our deep learning system, we utilize the K-fold cross-validation technique, which is beneficial for small datasets. The results show that the machine learning model acts as an excellent non-linear interpolator for DNS cases well-represented in the training set. In the most successful case, the L-infinity modeling error on the velocity profile was reduced from 23.4% to 4.0%. It is concluded that the developed machine learning methodology corresponds to a valid alternative to improve RANS turbulence models in flows with strong variations in their thermophysical properties without introducing prior modeling assumptions into the system.

This chapter has been published in Computers & Fluids, volume 255, p. 105835, 2023.

3.1. INTRODUCTION

3.1.1. TURBULENCE MODELLING

The governing equations of fluid flow have long been established, yet modelling turbulence remains one of the biggest challenges in engineering and physics. While it is possible to resolve the smallest scales of turbulent flows using direct numerical simulations (DNS), DNS is still unfeasible for real-world engineering applications. Due to this reason, engineers must rely on RANS turbulence models to describe turbulent flows. However, most of the development of turbulence models has focused on isothermal incompressible fluids. Therefore, these models can be inaccurate when applied to flows with strong variations in their thermophysical properties (He et al., 2008; Rodriguez et al., 2018), such as supercritical fluids or hypersonic flows. Understanding the behaviour of flows subject to strong property gradients is critical for several engineering applications, such as heat exchangers, supersonic aircraft, turbomachinery, and various applications in the chemical industry (Nemati et al., 2015; Pecnik & Patel, 2017; J. W. R. Peeters et al., 2016; J. Peeters, 2022; Yoo, 2013). Even incompressible fluids, such as water, can present large changes in viscosity when subjected to temperature variations.

For incompressible constant-property flows, the governing parameter in the description of turbulent boundary layers is the Reynolds number. For compressible flows, the Mach number and the associated changes in properties become additional parameters that characterize turbulent wall-bounded flows. From past studies, it is known that differences between a supersonic and a constant-property flow can be explained by simply accounting for the mean fluid property variations, as long as the Mach number remains small (Smits & Dussauge, 2006). This result is known as Morkovin's hypothesis (Morkovin, 1961). DNS of compressible channel flows (Coleman et al., 1995) also suggest that in the near-wall region most of the density and temperature fluctuations are the result of solenoidal 'passive mixing' by turbulence. Previous work by (Patel et al., 2015) has provided a mathematical basis for the use of the semi-local scaling as proposed by (Huang et al., 1995). It was concluded that under the limit of small property fluctuations in highly turbulent flows, a change in turbulence is governed by wall-normal gradients of the semi-local Reynolds number, defined as

$$Re_\tau^\star \equiv \frac{\sqrt{\overline{\rho}/\overline{\rho}_w}}{\overline{\mu}/\overline{\mu}_w} Re_\tau, \tag{3.1}$$

where ρ is the density, μ dynamic viscosity, the bar denotes Reynolds averaging, the subscript w indicates the value at the wall, and Re_τ is the friction Reynolds number based on wall quantities and the half channel height, h. Thus, $Re_\tau^\star$ provides a scaling parameter which accounts for the influence of variable properties on turbulent flows.

With the semi-local scaling framework and the fact that variable property turbulent flows can be successfully characterized by $Re_\tau^\star$, two main developments followed. First, in (Patel et al., 2016), a velocity transformation was proposed which allows to collapse mean velocity profiles of turbulent channel flows for a range of different density and viscosity distributions. Although following a different approach, this transformation is equivalent to the one proposed by (Trettel & Larsson, 2016). Second, this insight has later been used in (Pecnik & Patel, 2017) to extend the semi-local scaling framework

to derive an alternative form of the turbulent kinetic energy (TKE) equation. It was shown that the individual budget terms of this semi-locally scaled (TKE) equation can be characterized by the semi-local Reynolds number and that effects, such as solenoidal dissipation, pressure work, pressure diffusion and pressure dilatation, are indeed small for the flows investigated. Based on the semi-locally scaled TKE equation, (Rodriguez et al., 2018) derived a novel methodology to improve a range of eddy viscosity models. The major difference of the new methodology, compared to conventional turbulence models, is the formulation of the diffusion term in the turbulence scalar equations.

While these corrections improve the results of RANS turbulence models significantly, they can still be subject to further improvements. Due to these reasons, the present investigation will focus on building ML models to improve the performance of existing RANS turbulence models.

3.1.2. MACHINE LEARNING

In recent years, machine learning has been successfully applied in fluid mechanics and heat transfer due to its inherent ability to learn from complex data, see for instance (Chang et al., 2018). While different ML methods are available, deep neural networks have emerged as one of the most promising alternatives to improve turbulence modelling (Brunton et al., 2020). These systems are able to approximate complex non-linear functions by using nested layers of non-linear transformations, which can be adapted to the context of every application to optimize the usage of computational resources and to mitigate over-fitting. Different types of neural networks currently hold the state-of-the-art accuracy record in challenging domains, such as computer-vision or natural-language processing (LeCun et al., 2015). During the last decade, one of the main reasons behind the success of deep learning has been the ability of neural networks to approximate general non-linear functions while still providing multiple alternatives to optimize their design.

Significant works in the context of deep learning applied to CFD can be found in the studies of (Ling et al., 2016), who developed deep neural networks to model turbulence with embedded Galilean invariance, or in the work of (Parish & Duraisamy, 2016), where field inversion machine learning (FIML) is proposed in the context of CFD. Despite the abundance of recent works, significant research is still required regarding the application of ML in the context of CFD, and rich datasets to study turbulence in complex conditions must still be outlined. The future availability of datasets to study turbulence in complex geometries is particularly promising, as this could yield new models with strong applications to industrial and environmental problems.

The methodology for the present study is based on the FIML framework proposed by (Parish & Duraisamy, 2016). This methodology focuses on building corrections for existing RANS turbulence models instead of attempting to rebuild existing knowledge entirely. In the FIML framework, the process of building machine learning models is split into two stages. In the first stage, a data gathering process known as field inversion is performed, where the objective is to identify an ideal set of corrections for the RANS turbulence model under study. Then, in the second stage, a machine learning system is trained in order to replicate the corrections identified. The main advantage of this procedure is that the training process of a neural network is effectively decoupled from

the CFD solver, thereby improving the efficiency of the procedure by several orders of magnitude.

For the present work, several modifications are proposed with respect to the study made by (Parish & Duraisamy, 2016) and the subsequent publications of (Singh, Duraisamy, & Zhang, 2017; Singh, Matai, et al., 2017; Singh, Medida, & Duraisamy, 2017). The modifications considered cover different stages of the problem; such as the optimization methods employed in field inversion, the generation of automatic formulas to compute the gradients of the CFD system, the possibility to automate the process of generating feature groups for the ML system, and novel methods to improve the stability of the FIML methodology while making predictions.

3.2. FULLY DEVELOPED TURBULENT CHANNEL FLOWS

In this work we consider fully developed turbulent channel flows for which a large number of available DNS studies exist, and for which the time and space averaged conservation equations can be substantially simplified.

3.2.1. DNS DATABASE

The DNS database of turbulent planar channel flows, which we will consider, consists of three different sets of simulations. The first set represents variable property low-Mach number channel flows with isothermal walls, heated by a uniform volumetric source to induce an increase of temperature within the channel (Patel et al., 2016; Patel et al., 2017; Pecnik & Patel, 2017). Using different constitutive relations for viscosity μ, density ρ and thermal conductivity λ as a function of temperature, different DNS cases are used to study the effect of varying local Reynolds and Prandtl number on near wall turbulence. The cases with their respective relations for the transport properties and their corresponding wall-friction velocity based Reynolds number and local Prandtl number are summarized in table 3.1 (low-Mach number cases). Most of the cases have a friction based Reynolds number at the wall of Re_τ=395. Depending on the distribution of density, viscosity, and conductivity, the semi-local Reynolds number $Re_\tau^\star$ and the local Prandtl number are either constant, increasing or decreasing from the walls to the channel center. More details on the cases can be found in Refs. (Patel et al., 2016; Patel et al., 2017; Pecnik & Patel, 2017). The second set of DNS consists of high-Mach number compressible channel flow simulations with air modeled as a calorically perfect gas (Trettel & Larsson, 2016) (high-Mach number cases). The Mach number ranges from 0.7 to 4 and the corresponding constitutive laws for the transport properties, Re_τ and Prandtl number Pr are summarized in Table 3.1 as well. The third set of simulations contains incompressible channel flows (Jiménez & Hoyas, 2008) (incompressible cases). These cases been added as an additional set to train the FIML framework to account for a large range in Reynolds numbers.

For all of the variable property DNS cases, it is possible to show that Morkovin's hypothesis applies (Patel et al., 2015). This hypothesis establishes that only the averaged values in thermophysical properties can be used to characterize the changes in turbulence, and that any higher-order correlations of turbulent fluctuations observed in these properties have a negligible impact in the mean balances (Coleman et al., 1995; Patel et al., 2015).

Table 3.1: DNS database of turbulent channel flows with variable properties (low-Mach)(Patel et al., 2016; Pecnik & Patel, 2017), with ideal gases at high-Mach numbers (Trettel & Larsson, 2016), and with constant properties (incompressible) (Jiménez & Hoyas, 2008).

Number	Case ID	ρ	μ	λ	$Re_{\tau,w}$	Pr_w	$Ec_{\tau,w}$	ϕ
Low-Mach number cases (Patel et al., 2016; Patel et al., 2017; Pecnik & Patel, 2017)								
1	$CP150$	1	1	1	150			0
2	$CP395$	1	1	1	395			17.55
3	$CRe^{\star}_{\tau}$	T^{-1}	$T^{-0.5}$	1	395			17.55
4	$SRe^{\star}_{\tau GL}$	1	$T^{1.2}$	1	395			18.55
5	GL	T^{-1}	$T^{0.7}$	1	395			17.55
6	$LL1$	1	T^{-1}	1	150			29
7	$SRe^{\star}_{\tau LL}$	$T^{0.6}$	$T^{-0.75}$	1	150			31.5
8	$SRe^{\star}_{\tau Cv}$	1	$T^{-0.5}$	1	395	1	0	17.55
9	Cv	T^{-1}	T^{-1}	1	395			16
10	$LL2$	1	T^{-1}	1	395			17.55
11	$CRe^{\star}_{\tau}CPr^{\star}$	T^{-1}	$T^{-0.5}$	$T^{-0.5}$	395			17.55
12	$GLCPr^{\star}$	T^{-1}	$T^{0.7}$	$T^{0.7}$	395			17.55
13	$V\lambda SPr^{\star}_{LL}$	1	1	T^{1}	395			17.55
14	$JFM.CRe^{\star}_{\tau}$	T^{-1}	$T^{-0.5}$	1	395			95
15	$JFM.GL$	T^{-1}	$T^{0.7}$	1	950			75
16	$JFM.LL$	1	T^{-1}	1	150			62
High-Mach number cases (Trettel & Larsson, 2016)								
17	$M0.7R400$				437		$5.736{\cdot}10^{-4}$	
18	$M0.7R600$				652		$5.190{\cdot}10^{-4}$	
19	$M1.7R200$				322		$2.804{\cdot}10^{-3}$	
20	$M1.7R400$				663		$2.394{\cdot}10^{-3}$	
21	$M1.7R600$	$\propto T^{-1}$	$T^{0.75}$	$T^{0.75}$	972	0.7	$2.135{\cdot}10^{-3}$	0
22	$M3.0R200$				650		$4.751{\cdot}10^{-3}$	
23	$M3.0R400$				1232		$4.185{\cdot}10^{-3}$	
24	$M3.0R600$				1876		$3.752{\cdot}10^{-3}$	
25	$M4.0R200$				1017		$5.574{\cdot}10^{-3}$	
Incompressible cases (Jiménez & Hoyas, 2008)								
26	$IC.Re180$				180			
27	$IC.Re550$				550			
28	$IC.Re950$	-	-	-	950	-	-	-
29	$IC.Re2000$				2000			
30	$IC.Re4200$				4200			

3.2.2. RANS EQUATIONS

To model the turbulent channel flows described above, we use the Reynolds/Favre averaged Navier-Stokes equations. For a fully developed turbulent channel flow, the only in-homogeneous direction of the averaged flow corresponds to the wall-normal coordinate, leading to a set of one-dimensional partial differential equations for the mean momentum, mean energy and any additional transport equations for the turbulence quantities used to close the RANS equations. The Reynolds/Favre averaged streamwise momentum and energy equations for a fully developed turbulent channel flow read

$$\frac{\partial}{\partial y}\left[\left(\frac{\mu}{Re_{\tau,w}}+\mu_t\right)\frac{\partial u}{\partial y}\right]=-1, \tag{3.2}$$

$$\frac{\partial}{\partial y}\left[\left(\frac{\lambda}{Re_{\tau,w}Pr_w}+\frac{c_p\mu_t}{Pr_t}\right)\frac{\partial T}{\partial y}\right]=-Ec_{\tau,w}\left(\frac{\mu}{Re_{\tau,w}}+\mu_t\right)\left(\frac{\partial u}{\partial y}\right)^2-\frac{\phi}{Re_{\tau,w}Pr_w}, \tag{3.3}$$

with the variables u and T referring to the Favre-averaged streamwise velocity and the cross-sectional temperature profiles, respectively. The variables c_p, Pr_t and ϕ refer to the isobaric heat capacity, the turbulent Prandtl number, and an arbitrary volumetric heat source term. The coordinates x and y further refer to the streamwise and the wall-normal directions for the channel flow. The wall based friction Reynolds number, the Prandtl number and the friction based Eckert number are defined as

$$Re_{\tau,w}=\frac{\rho_w u_{\tau,w} h}{\mu_w},\quad Pr_w=\frac{\mu_w c_{p,w}}{\lambda_w},\quad Ec_{\tau,w}=u_{\tau,w}^2/(c_{p,w}T_w)=\left(\gamma-1\right)Ma_{\tau,w}^2, \tag{3.4}$$

with $u_{\tau,w}=\sqrt{\tau_w/\rho_w}$ the friction velocity, h the channel half width, γ the ratio of specific heats and $Ma_{\tau,w}=u_{\tau,w}/a_w$, where τ_w is shear stress and a_w the speed of sound at the wall. Given these non-dimensional groups, the non-dimensional density, temperature, viscosity, thermal conductivity are one at the wall, while the non-dimensional isobaric heat capacity is $c_p=1$ in the whole domain.

The mean momentum and energy equations make use of the Boussinesq approximation and the strong Reynolds analogy to model the turbulent shear stress, the turbulent heat transfer, and the turbulent dissipation in the energy equation (first term on the right-hand-side). As such, a turbulent eddy viscosity μ_t appears in eqs. (3.2) and (3.3), which is commonly provided by an eddy viscosity model. While many eddy viscosity models exist in literature, in this work we choose the Myong-Kasagi $k-\varepsilon$ turbulence model (MK) (Myong & Kasagi, 1990), which has also been used in our previous studies to model turbulence in variable property turbulent channel flows (Rodriguez et al., 2018). The equations for the turbulent kinetic energy k and turbulent dissipation ε read

$$\underbrace{\mu_t\left(\frac{\partial u}{\partial y}\right)^2}_{P_k}-\underbrace{\rho\,\varepsilon}_{D_k}+\underbrace{\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial y}\right]}_{T_k}=0, \tag{3.5}$$

$$\underbrace{C_{\varepsilon 1}\,P_k\,\frac{\varepsilon}{k}}_{P_\varepsilon}-\underbrace{C_{\varepsilon 2}\,f_\varepsilon\,\rho\,\frac{\varepsilon^2}{k}}_{D_\varepsilon}+\underbrace{\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial y}\right]}_{T_\varepsilon}=0, \tag{3.6}$$

with the supporting damping functions

$$f_\varepsilon = \left(1 - \frac{2}{9} e^{-(Re_t/6)^2}\right)\left(1 - e^{-y^\star/5}\right)^2, \quad f_\mu = \left(1 - e^{-y^\star/70}\right)\left(1 + \frac{3.45}{\sqrt{Re_t}}\right), \tag{3.7}$$

and the definition of the turbulent Reynolds number, the semi-locally scaled wall distance and the eddy viscosity, respectively,

$$Re_t = \frac{\rho\, k^2}{\mu\, \varepsilon}, \quad y^\star = y^+ \sqrt{\frac{\rho}{\rho_w}} \frac{\mu_w}{\mu}, \quad \mu_t = C_\mu\, f_\mu\, \rho\, \frac{k^2}{\varepsilon}. \tag{3.8}$$

The constants take the following values: $C_{\varepsilon 1} = 1.4$, $C_{\varepsilon 2} = 1.8$, $C_\mu = 0.09$, $\sigma_k = 1.4$ and $\sigma_\varepsilon = 1.3$. Note, the original model uses the wall distance based on viscous wall units y^+ in the damping functions. Here, we replaced y^+ with $y^\star$ to account for the changes in viscous length scales due to changes in density and viscosity close to the wall (Rodriguez et al., 2018). The turbulent Prandtl Pr_t is set to unity in all cases. For the high-Mach number cases, a detailed analysis showed that $Pr_t \approx 1$ in the buffer layer, where the largest turbulent heat fluxes can be found. Similarly, (Patel, 2017) found that $Pr_t \approx 1$ in the buffer layer for the low-Mach number cases in our database. The Python and the Matlab source codes to solve the set of RANS equations with the associated boundary conditions can be found on Github (Pecnik et al., 2018).

The velocity profiles for a few selected cases with the original MK turbulence model are shown in Fig. 3.1. Large deviations occur in flows subject to strong variable-property gradients. The largest deviations found in such regimes can be found in the DNS case $JFM.CRe_\tau^\star$ from Table 3.1. Here, it can be noted that the maximum error margin reaches a magnitude of 22.8% at the channel center ($y = H$). Based on these results, it can be noted that the MK turbulence model corresponds to an interesting target for ML optimization, since there exist large deficits to be mitigated.

3.3. IMPROVED FIELD INVERSION MACHINE LEARNING METHODOLOGY FOR VARIABLE PROPERTY TURBULENCE

In this section we present an improved methodology of the FIML as proposed by (Parish & Duraisamy, 2016), which is also suitable to account for turbulence in variable-property flows.

3.3.1. FIELD INVERSION

In order to minimize the difference between the DNS and the modeled velocity obtained with the RANS approach, the original $k-\varepsilon$ equations are modified by introducing field inversion multipliers β. The turbulent kinetic energy k and the turbulent dissipation ε, eqs. (3.5) and (3.6), can then be written as

$$P_k - \beta_k D_k + T_k = 0, \tag{3.9}$$

$$P_\varepsilon - \beta_\varepsilon D_\varepsilon + T_\varepsilon = 0. \tag{3.10}$$

Contrary to (Parish & Duraisamy, 2016), we multiply the dissipation rather than the production terms, in order to adhere to energy conservation, i.e. turbulent kinetic production

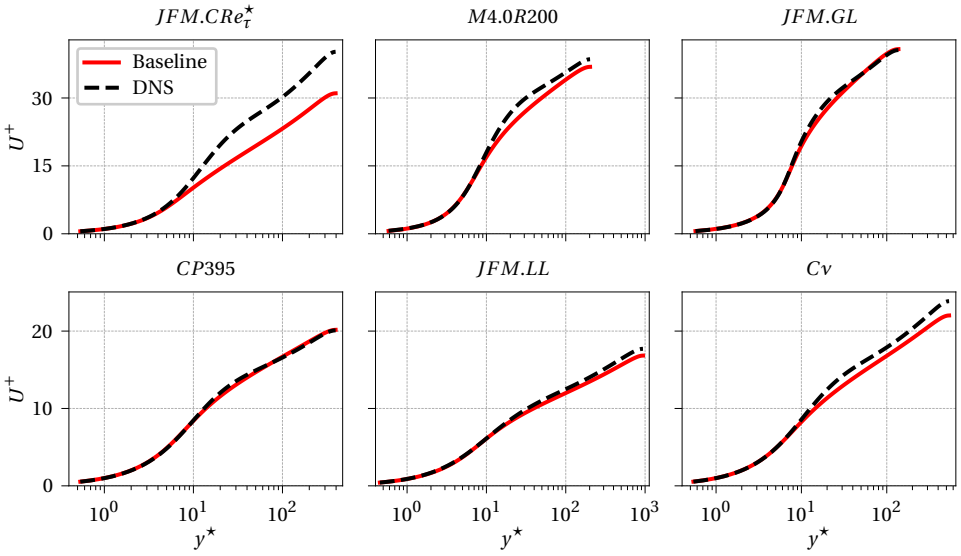


Figure 3.1: Velocity profiles obtained with the original MK turbulence model using the density and viscosity profiles obtained from DNS. The black dashed lines correspond to the DNS data, whereas the red solid lines correspond to the RANS simulations.

also appears in the mean kinetic energy equation with an opposite sign (Durbin & Reif, 2011). On the other hand, introducing β_k in the k-equation can lead to an imbalance of turbulent production and dissipation in the log-layer region. Therefore, we also present results where only β_ε is used to perform the field inversion, since the ε-equation contains the largest amount of empiricism. Another reason to modify the dissipation instead of the production terms is because the production is active in a smaller region of turbulent boundary layers. This implies that field inversion optimizers modifying the dissipation term have a larger capacity to build corrections in regions where other budget terms of RANS turbulence models are still active, such as the diffusion terms.

It is important to note that field inversion optimizers build corrections, which are ideal with respect to the cost function formulated. As a result, the cost function for the field inversion process must be carefully designed, to minimize not only the differences between the velocity profiles, but also the shape of the corrections that will be applied to the turbulence model. A suitable cost function $\mathscr{J}$ is defined as

$$\mathscr{J} = \sum_{i=1}^{N} I_U \left(\frac{u_i - u_i^*}{S_U} \right)^2 + I_k \left(\frac{\delta_k}{S_k} \right)^2 + I_\varepsilon \left(\frac{\delta_\varepsilon}{S_\varepsilon} \right)^2, \tag{3.11}$$

with individual weights I_U, I_k and I_ε for each term in the cost function. The first term represents the difference between the RANS velocity profiles (u) and the DNS data (u^*), whereas the subsequent terms are equivalent to source/sink terms in the turbulence

modelling equations. It can easily be shown that δ is related to β as

$$\delta_k = D_k\left(\beta_k - 1\right), \tag{3.12}$$

$$\delta_\varepsilon = D_\varepsilon\left(\beta_\varepsilon - 1\right). \tag{3.13}$$

Finally, S_U, S_k and S_ε are used to normalize the variations in the cost function, and they are defined as

$$S_U = max(\left|u^*\right|), \tag{3.14}$$

$$S_k = max\left(|P_k|,\, |D_k|,\, |T_k|\right), \tag{3.15}$$

$$S_\varepsilon = max\left(|P_\varepsilon|,\, |D_\varepsilon|,\, |T_\varepsilon|\right). \tag{3.16}$$

δ_k and δ_ε are normalized such that the importance factors, I, are easier to interpret among all DNS cases considered in this study. As it can be seen in the present formulation, the final field inversion study must include an hyper-parameter optimization analysis for the values of I_U, I_k and I_ε. The selection method is based on the elbow method (Thorndike, 1953), since it was found that each field inversion shows clear inflection points (discussed in detail later).

OPTIMIZATION ALGORITHM

To solve the field inversion problems, we will use gradient-descent (GD) algorithms. In general, GD algorithms are preferred over Hessian methods to solve complex non-linear optimization problems across different fields. Moreover, for our specific application, it can be shown that the Hessian matrix is non-invertible at the channel center due to the vanishing gradients near the symmetry plane. Accordingly, the β multipliers at the channel center have a negligible effect on the solution, and thus their influence on the cost function $\mathscr{J}$ is nearly zero. Another favorable property of GD algorithms is that their results yield continuous spatial distributions due to the smoothness of the gradients associated with the turbulence model. Furthermore, GD algorithms tend to leave the β multipliers near the channel center at their initial values ($\beta = 1$), since these algorithms do not modify parameters which are not relevant to the cost function $\mathscr{J}$.

The GD algorithm used in the present study is based on the traditional bold drive method (Battiti, 1989). However, we also introduced gradient inertia to increase the convergence speed. The final approach for the optimizer is shown in algorithm 3. In this algorithm, the optimizer starts by taking a traditional step using gradient-descent with added momentum. The values generated for the gradient inertia and the optimization parameters are stored using the auxiliary variables m' and β' respectively. If the updated value for the cost function $\mathscr{J}\left(\beta'\right)$ is lower than before, the temporary values for m' and β' are accepted as the new state of the system. Additionally, the learning rate α is increased according to the expansion ratio k^+. This allows the optimizer to dynamically search for a learning rate schedule that maximizes the convergence speed. If divergence is detected ($\mathscr{J}\left(\beta'\right) > \mathscr{J}_{n-1}$), the optimizer retains the β parameters from the previous iteration (β_{n-1}), resets the gradient inertia (m) to the current Jacobian, and decreases the learning rate according to the ratio k^-. These simple steps allow the optimizer to perform a line-search process, seeking optimal values for the learning rate α. Gradient inertia must be

Algorithm 3 Modified bold drive method with added momentum to accelerate optimization.

while $\alpha_{n-1} >$ Threshold **do**
  $m' \leftarrow c \cdot m_{n-1} + (1-c)\nabla_\beta \mathscr{J}_{n-1}$
  $\beta' \leftarrow \beta_{n-1} - \alpha_{n-1} \cdot m'$
  if $\mathscr{J}(\beta') < \mathscr{J}_{n-1}$ **then**
    $\beta_n \leftarrow \beta'$
    $m_n \leftarrow m'$
    $\alpha_n \leftarrow k^+ \cdot \alpha_{n-1}$
  else
    $\beta_n \leftarrow \beta_{n-1}$
    $m_n \leftarrow \nabla_\beta \mathscr{J}_{n-1}$
    $\alpha_n \leftarrow k^- \cdot \alpha_{n-1}$
  end if
end while

necessarily removed from the system during the line-search process, since otherwise it cannot be guaranteed that the algorithm will converge to an optimized β distribution.

The recommended values from literature for the parameters k^+ and k^- are 1.1 and 0.5, respectively. However, in the present study, we employ a more aggressive expansion value of $k^+ = 1.2$. The constant c corresponds to the gradient inertia hyper-parameter. For this variable, a recommended value of $c = 0.9$ can be found across a wide variety of algorithms described in the literature (Kingma & Ba, 2015; Mitliagkas et al., 2016). It was found that the introduction of the gradient inertia decreased the running times by a factor three with respect to the original bold drive method. The proposed algorithm allows to fully automatize the process of field inversion, and to subsequently run over 450 optimization cases in total.

JACOBIAN MATRIX CALCULATION

The Jacobian associated with the field inversion process is computed using the discrete adjoint method. In this method, the discretized RANS equations are written as a residual vector $\mathscr{R}(W(\beta),\beta) = 0$, that contains one entry per every discretized cell and scalar equation. The variable $W(\beta)$ corresponds to the vector of dicretized degrees of freedom present in the RANS equations, such as the velocities (u) or the turbulent scalar quantities k and ε. According to the discrete adjoint method, the Jacobian $\nabla_\beta \mathscr{J}$ can be calculated as

$$\nabla_\beta \mathscr{J} = \Psi^T \cdot \frac{\partial \mathscr{R}}{\partial \beta} + \frac{\partial \mathscr{J}}{\partial \beta}, \tag{3.17}$$

where the vector Ψ can be obtained from the following system of linear equations

$$\left[\frac{\partial \mathscr{R}}{\partial W}\right]^T \cdot \Psi = -\left[\frac{\partial \mathscr{J}}{\partial W}\right]^T. \tag{3.18}$$

The main advantage of the discrete adjoint method is that only the vector Ψ must be calculated from Eq. (3.18), whereas a direct calculation method based on chain-rule

differentiation would require the computation of the rank 2 sensitivity matrix $\partial W/\partial\beta$. Since the latter matrix is orders of magnitude larger than the vector Ψ, the discrete adjoint method constitutes a better alternative.

In order to generate explicit formulas for all the entries present in the matrices $\partial\mathscr{R}/\partial W$ and $\partial\mathscr{R}/\partial\beta$, we utilize symbolic algebra packages, such as Sympy (Meurer et al., 2017). As a result, the coefficients of these matrices are described by long arithmetic formulas, which can be inserted into the source code of a function written in any programming language. The use of explicit formulas increases the speed of our optimizer as any zero coefficients are immediately cancelled by the algebraic package. Moreover, commonly repeated algebraic sub-terms, such as the eddy viscosity $\mu_t = C_\mu\ f_\mu\ \rho\ k^2/\varepsilon$, can be replaced by auxiliary variables to avoid redundant calculations.

3.3.2. NEURAL NETWORKS

In order to complete the FIML methodology, we construct a predictive system using neural networks. Our neural networks utilize hyperbolic tangent neurons in their deeper layers, due to their inherent ability to produce smooth output distributions and since they fulfill the universal approximation theorem (Cybenko, 1989; Hornik, 1991). An early reference to the use of hyperbolic tangent neurons in the context of fluid mechanics can be found in the work of (Milano & Koumoutsakos, 2002). In the first layer of our neural networks we introduce logarithmic neurons (Hines, 1996), which reduce the dimensionality of the input features, identifying the best parameter groups relevant to a regression problem. Therefore, the introduction of logarithmic neurons in neural networks allows the optimizer to determine which feature groups are optimal in the context of fluid mechanics, even in the absence of previous modelling knowledge.

All the neural networks trained during the current study are based on a mean-squared error (MSE) loss function for the δ corrections, plus an additional L2 regularization term for the weights w in the neural network:

$$\mathscr{J}_{train} = \frac{1}{N}\underbrace{\sum_{i=1}^{N}\left(\delta_{NN,i}-\delta_{FI,i}\right)^2}_{\|\delta_{error}\|^2} + \lambda\ \frac{1}{M}\underbrace{\sum_{j=1}^{M} w_j^2}_{\|w\|^2}. \tag{3.19}$$

In Eq. (3.19), the variables δ_{NN} and δ_{FI} correspond to the corrections predicted by each neural network and the reference field inversion data, respectively. The hyper-parameter λ corresponds to a constant which must be calibrated to mitigate over-fitting in the system. During the current study, the values for λ were calibrated by applying the elbow method to the training datasets exclusively, without considering external cross-validation datasets. This is possible, since the inflection point in the residual errors $\|\delta_{error}\|$ with respect to the training data can be tracked to establish the magnitude at which λ is able to produce changes, and likely mitigate over-fitting. Therefore, all DNS cases which are not included in the training set of a neural network can be considered as purely held-back test cases.

Beyond reducing over-fitting, another important consequence of introducing L2 regularization is that the final neural networks will assign small weights, and thus low importance, to the input features X_i which do not facilitate the regression process. Therefore, the feature importance rankings generated after using L2 regularization will display

more consistent trends regarding the most valuable features to perform predictions. The methodology used to rank the importance of every feature in the neural networks is presented later in Section 3.3.4.

K-FOLD VALIDATION

Due to the relatively small size of our database for the machine learning procedure, in combination with the diversity of cases, it proved difficult to split the data between training, cross-validation (CV) and test sets. Picking relevant CV sets that were unbiased by prior turbulence modelling knowledge is difficult, since the uniqueness of many DNS samples contained in our database implied that the cases picked for cross-validation could greatly underestimate the error margins found in the test set. As a result, employing CV sets proved to be ineffective. Therefore, the study was performed using the K-fold validation method (Mosteller & Tukey, 1968). This method assesses the robustness of machine learning models by picking "K" random training sets, and subsequently evaluating the results with the remaining test set. If a large variance is detected among the K-fold trials, this may indicate that more data is required to train the ML system effectively, or that a different ML architecture is required. The K-fold validation method corresponds to one of the best alternatives available to assess the performance of ML systems trained with small datasets (Nie et al., 2020).

The test sets for the different K-fold combinations are listed in Table 3.2. The DNS cases for testing the machine learning framework are picked randomly, except for the K-1 set. The test set K-1 contains cases with the most extreme property variations, such as $JFM.CRe_\tau^\star$ and $M4.0R200$. As a result, the K-1 set represents a scenario where challenging predictions are required, despite the absence of adequate training samples. The incompressible DNS cases from the work of (Jiménez & Hoyas, 2008) are added to the test sets of the K-fold validation trials (K-2 to K-10) in order to assess the response of the ML system for different Reynolds numbers. Each trial in the K-fold methodology is an independent machine learning study, with five or six completely unknown test cases for model validation; see Table 3.2. All the hyper-parameters in the model were calibrated by using only the training set in conjunction with the elbow method. The test cases were not used for the calibration of any of hyper-parameter in the study.

The selection procedure to determine the final machine model for the study is based on finding the smallest neural network architecture which is capable of fitting the training data available for the K-fold combination (K-1) listed in Table 3.2. According to the principles of the elbow method, the smallest system which can fit the training data is less likely to produce over-fitting than larger machine learning models. The K-fold set (K-1) is chosen, since this combination represents a realistic scenario where a selection of the most challenging CFD cases remain hidden from the training data. In summary, the neural network architecture is not pre-conditioned to perform well under the most complex test conditions available.

WEIGHTED RELAXATION FACTOR

In a preliminary analysis of the ML methodology, spurious oscillations could occur in the predicted δ corrections. Such oscillations could result in numerical instabilities in the CFD-solver. To avoid this behaviour, a novel weighted relaxation factor method is introduced, which is able to filter spurious oscillations in the predicted δ corrections. To

Table 3.2: Configuration considered during the implementation of the K-fold validation methodology. All test cases are marked with checkmarks.

Case ID	K-1	K-2	K-3	K-4	K-5	K-6	K-7	K-8	K-9	K-10
Low-Mach number cases										
$CP150$				✓					✓	
$CP395$					✓				✓	✓
$CRe_{\tau}^{\star}$										
$SRe_{\tau GL}^{\star}$								✓		
GL							✓			
$LL1$					✓					
$SRe_{\tau LL}^{\star}$		✓						✓		
$SRe_{\tau Cv}^{\star}$.										✓
Cv	✓				✓	✓				
$LL2$				✓						
$CRe_{\tau}^{\star}CPr^{\star}$	✓					✓	✓			✓
$GLCPr^{\star}$		✓					✓		✓	
$V\lambda SPr_{LL}^{\star}$				✓				✓		
$JFM.CRe_{\tau}^{\star}$	✓	✓	✓	✓			✓			✓
$JFM.GL$			✓			✓		✓		
$JFM.LL$			✓		✓				✓	
High-Mach number cases										
$M0.7R400$					✓	✓				✓
$M0.7R600$										
$M1.7R200$		✓					✓			
$M1.7R400$								✓		
$M1.7R600$									✓	
$M3.0R200$	✓			✓						
$M3.0R400$										
$M3.0R600$			✓							
$M4.0R200$	✓	✓	✓			✓				
Incompressible cases										
$IC.Re180$									✓	
$IC.Re550$			✓		✓		✓			
$IC.Re950$		✓		✓						✓
$IC.Re2000$						✓				
$IC.Re4200$								✓		

explain this, we show in Fig. 3.2 the turbulent kinetic energy budgets (P_k, D_k, T_k) and three profiles of δ corrections for the DNS case $JFM.CRe_\tau^\star$. The variable δ_{FI} presents the ground-truth labels obtained through field inversion, whereas δ_{ini} corresponds to a fictitious set of corrections with added noise in the form of $\Delta = 0.6\ y^3\ sin\left(8\pi y\right)$. The goal of the weighted relaxation factor is to obtain corrections δ_{ML} that closely represent δ_{FI}, without any significant oscillations.

The derivation of the weighted relaxation factor starts by noting that the magnitude of the final corrections that will be applied to the RANS model, δ_{ML}, only corresponds to a fraction, α, of the original corrections predicted by a neural network, δ_{ini}. This relation can be stated as

$$||\delta_{ML}|| = \alpha\ ||\delta_{ini}||, \tag{3.20}$$

or alternatively,

$$\mathscr{J}_\delta = \sum_{i=1}^{N} \delta_{ML,i}^2 = \sum_{i=1}^{N} \left(\alpha\ \delta_{ini,i}\right)^2\ (= constant). \tag{3.21}$$

In order to assess the true compatibility of the final δ_{ML} corrections with a given RANS turbulence model, we express δ_{ML} in eq. (3.21) as β times the production term P, namely

$$\delta_{ML} = \beta P \tag{3.22}$$

in the corresponding turbulence modelling equations. The key idea to build a robust methodology is to recognize that spurious machine learning corrections, such as δ_{ini} in Fig. 3.2, create large oscillations in the β multipliers defined by Eq. (3.22). As a result,

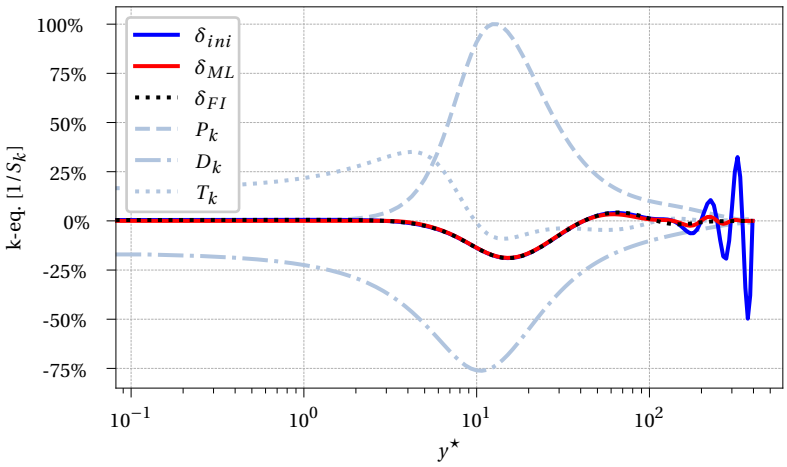


Figure 3.2: Turbulence budgets for the optimized k-equation of the MK turbulence model after applying the independent set of δ_k corrections obtained during the field inversion study for the DNS case $JFM.CRe_\tau^\star$. The initial machine learning predictions (δ_{ini}), shown in blue, contain the ficticious perturbation term: $\Delta = 0.6\ y^3\ sin\left(8\pi y\right)$. The red line δ_{ML} corresponds to the corrections obtained after applying the weighted relaxation factor methodology.

the introduction of a L2-regularization hyper-parameter, λ, for these β multipliers would immediately penalize the presence of large oscillations in regions where RANS turbulence models are inactive. The cost function associated with this problem is

$$\begin{aligned} \mathscr{J}_\beta &= \sum_{i=1}^{N} \left(\delta_{ML,i} - \delta_{ini,i}\right)^2 + \lambda \beta_i^2 \\ &= \sum_{i=1}^{N} \left(P_i \beta_i - \delta_{ini,i}\right)^2 + \lambda \beta_i^2. \end{aligned} \tag{3.23}$$

Eq. (3.23) states that the final β multipliers must produce the greatest degree of similarity between δ_{ML} and δ_{ini}, while minimizing the magnitude of $||\beta^2||$ according to a regularization hyper-parameter λ. In order to minimize the cost function defined in Eq. (3.23), its Jacobian can be forced to form a null vector:

$$\nabla_\beta \mathscr{J}_\beta = 0. \tag{3.24}$$

Replacing Eq. (3.23) into the previous condition, yields the following element-wise array equation:

$$P\left(P\beta - \delta_{ini}\right) + \lambda\beta = 0. \tag{3.25}$$

Re-arranging the terms of Eq. (3.25) further reveals that

$$\beta = \frac{\delta_{ini}\, P}{\lambda + P^2}. \tag{3.26}$$

Note, eq. (3.26) is evaluated element-wise. Replacing eq. (3.26) back into eq. (3.22) gives a direct residual equation for λ

$$\mathscr{R}_\lambda = \sum_{i=1}^{N} \left(\delta_{ini,i}\, \frac{P_i^2}{\lambda + P_i^2} \right)^2 - \sum_{i=1}^{N} \left(\alpha\, \delta_{ini,i}\right)^2 = 0. \tag{3.27}$$

Since Eq. (3.27) only contains one unknown (λ), a simple root-finding algorithm can be used to solve this optimization problem, such as the Newton–Raphson method. For reference, the gradient of the previous residual equation ($\mathscr{R}_\lambda$) is given by the following formula:

$$\nabla_\lambda \mathscr{R}_\lambda = -2 \sum_{i=1}^{N} \frac{\left(\delta_{ini,i} P_i^2\right)^2}{\left(\lambda + P_i^2\right)^3}. \tag{3.28}$$

After obtaining the regularization hyper-parameter, λ, the final ML corrections (δ_{ML}) are given by:

$$\delta_{ML} = \frac{\delta_{ini}\, P^2}{\lambda + P^2}. \tag{3.29}$$

Eqs. (3.27-3.29) constitute the only required components to implement our weighted relaxation factor methodology in a computer environment. The results depicted in Fig. 3.2, show clearly that the weighted relaxation factor method is able to filter the added noise. The final distribution obtained, effectively resembles the ground-truth labels, δ_{FI}, which were hidden from the system.

3.3.3. FINAL FRAMEWORK

The final machine learning framework for the study can be found in Fig. 3.3, which is split into two stages. In the first stage, shown in Fig. (3.3.a), DNS data is used to generate δ_{FI} field inversion corrections for each case, and to subsequently train neural networks that can predict the identified $\delta(y)$ distributions. The predictions of the neural networks are based on stacks of input features X_f extracted from the uncorrected version of the MK model ($\delta = 0$). One of the main differences between the framework described in Fig. (3.3.a) and the original approach proposed by Parish & Duraisamy (Parish & Duraisamy, 2016) is that our field inversion corrections δ are subject to L2-regularization based on their absolute magnitude as a fourth budget-term in the RANS equations, instead of their values as relative β multipliers with respect to existing RANS terms. This enables the field inversion optimizer to build corrections that follow patterns which are not captured by the baseline RANS models. Additionally, the framework described in Fig. 3.3(a) has been adapted to account for the changes observed in flows subject to strong variations in their thermophysical properties, namely, ρ, μ and λ. The approach effectively decouples the analysis of the RANS momentum equations from the energy equation or any associated equation-of-state for fluids. This is achieved by passing the DNS profiles for the density and dynamic viscosity to the baseline RANS turbulence models during the field inversion process. However, one challenge introduced by this procedure is that the stack of input features X_f to predict the field inversion corrections δ must be based on accurate estimations of the profiles for the thermophysical properties of fluids.

This challenge was solved in the second stage of the ML framework presented in Fig. 3.3(b), where an iterative feedback loop is used to create predictions for unknown CFD cases. At the start of this feedback loop, the uncorrected version of the MK turbulence model is solved, which yields an initial estimate for ρ and μ. Then, a stack of input features X_f is created to describe the behavior of the uncorrected MK model, and to subsequently generate neural network predictions for the optimal $\delta_{ML}(y)$ corrections. Before injecting these δ_{ML} corrections into a CFD solver, the weighted relaxation factor methodology described in Section 3.3.2 is applied. The final $\delta_{ML}(y)$ corrections are then inserted back into the MK turbulence model as an explicit source term ($+\,\delta_{ML}(y)$). The feedback loop described in Fig. 3.3(b) is completed by generating a new estimate for the thermophysical properties μ and ρ, and by repeating the previous steps until convergence is achieved.

The final neural network architecture is depicted in Fig. 3.4. The neural network contains only three logarithmic neurons in the initial layer and 77 trainable parameters. The initial stack of features, presented in Fig. 3.4, corresponds to different physical quantities that may be considered by the neural network. The previous quantities are intended to be computed based on the initial turbulence budgets found in the uncorrected RANS equations. The sub-scales M_k and M_ε correspond to references used to normalize the scalar fields k and ε based on the magnitude of the destruction terms in the RANS equations:

$$M_\varepsilon = \frac{S_k}{\rho_w}, \quad M_k = \frac{\rho_w M_\varepsilon^2}{S_\varepsilon}. \tag{3.30}$$

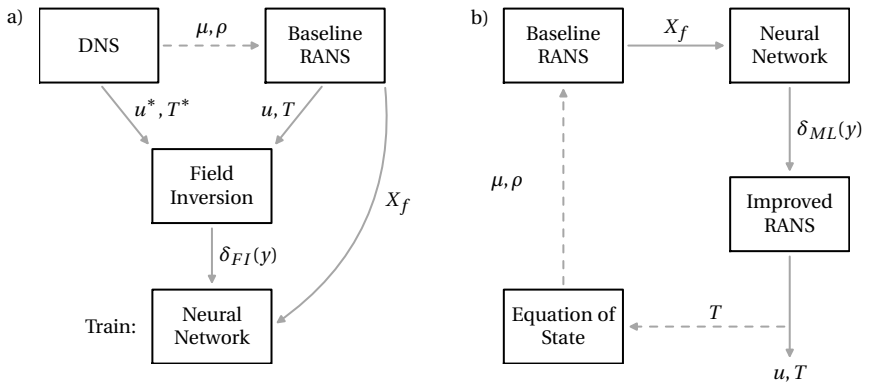


Figure 3.3: Final framework established for the FIML methodology. The dashed lines indicate the additional steps which are necessary to handle the presence of variable-property flows, with respect to the original scheme proposed by (Parish & Duraisamy, 2016). The diagram on the left (a) presents the methodology employed to obtain field inversion corrections and to train deep learning systems, whereas the scheme on the right (b) corresponds to the feedback loop used to perform predictions at runtime.

3.3.4. INTERPRETATION OF MACHINE LEARNING RESULTS

Regarding the interpretation of the final machine learning results, the integrated gradients (IG) (Sundararajan et al., 2017) was used to estimate the importance of every feature passed to the neural network shown in Fig. 3.4. This method performs a numerical integration for the gradients of the ML predictions ($\delta(X)$) with respect to the stack of input features X, following a linear path starting from a common baseline state X_0 (Martín Abadi et al., 2015; Sundararajan et al., 2017):

$$IG_i = (X_i - X_{i,0}) \int_{\alpha=0}^{1} \frac{\partial}{\partial X_i} \left[\delta(X_0 + \alpha(X - X_0))\right] d\alpha. \tag{3.31}$$

In eq. (3.31), the term IG_i corresponds to the importance score assigned to every feature X_i passed to the neural network. Here, X_0 represents the average values of every feature at each y-location $X_0 = X_0(y)$. The main benefit of this method is that the final scores are not subject to the sensitivity of the ML predictions with respect to infinitesimal changes in X_i, but rather represent the importance of global changes in the input features.

3.4. FIELD INVERSION RESULTS

This section will describe the results of the FIML study for the MK turbulence model. First, the different hyper-parameter combinations for the field inversion study will be analyzed. Then, the observed trends in the final machine learning predictions will be presented, followed by a brief discussion of the results.

The field inversion study of the MK turbulence model focuses on determining the values of the hyper-parameters I_U, I_k and I_ε in eq. (3.11). In the first combination, an equal importance is assigned to the corrections used in each scalar equation (k and ε) by setting $I_k = I_\varepsilon = 1$. The value of I_U was calibrated by applying the elbow method to

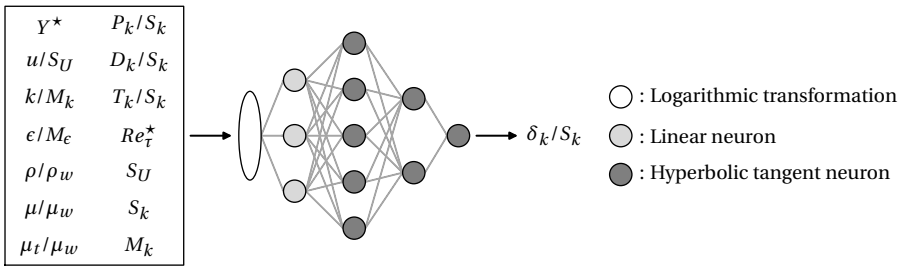


Figure 3.4: Neural network architecture created to predict the field inversion corrections (δ_k) required by the MK turbulence model.

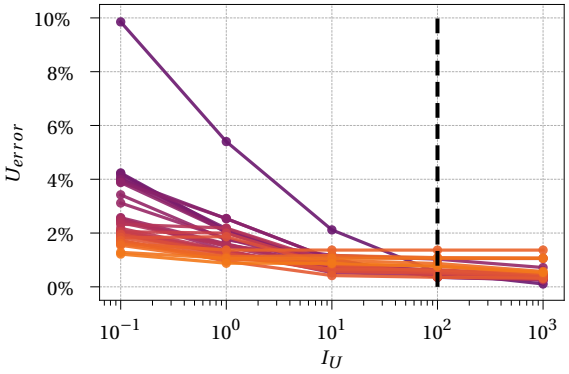


Figure 3.5: Application of the elbow method to determine the magnitude of I_U during the initial field inversion study for the MK turbulence model ($I_k = I_\varepsilon = 1$). The black dashed line represents the position where $I_U = 100$.

the system. The results obtained can be found in Fig. 3.5, where it can be seen that clear inflection points exist for each case. For any subsequent ML analysis, it is possible to either choose the I_U values located at the inflection point of each DNS case, or to pick a common value of I_U for all cases. It was decided to pick a common value of $I_U = 100$ for all DNS cases, since this creates smooth trends across the whole dataset. Additionally, selecting a unique value for I_U can simplify the creation of deep learning models, since all the target δ_{FI} corrections correspond to the solution of a single optimization problem. If different I_U values were picked for each DNS case, additional training data might be required to allow the deep learning system to approximate the selection criterion employed.

The second stage of the hyper-parameter optimization study consists in analyzing the effect of changing the individual values of I_k and I_ε in the field inversion results. The effects of varying these hyper-parameters are depicted in Fig. 3.6 for the DNS case $CRe_\tau^\star$, which corresponds to the case with the highest modelling errors using the MK turbulence model. The results show that different combinations for the values of I_k and I_ε yield similar shapes for the corrections, since only the magnitude of the peaks change. Moreover, even building independent sets of either δ_k or δ_ε corrections yields similar

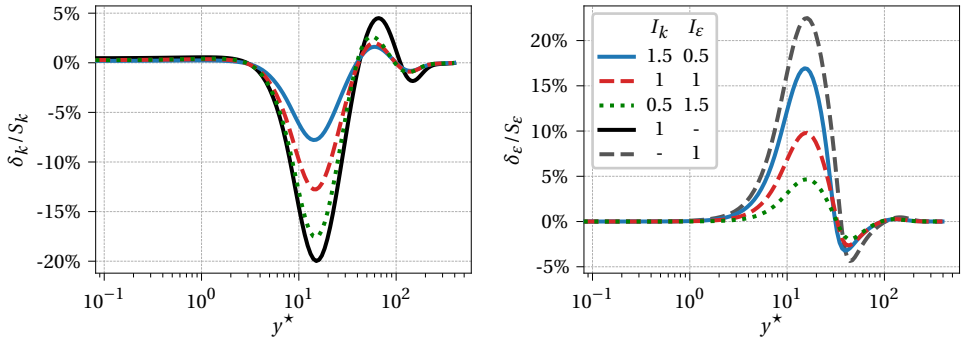


Figure 3.6: Effect of the cross-interactions between I_k and I_ε for $I_U = 100$ during the field inversion study for the MK turbulence model considering the DNS case $JFM.CRe_\tau^\star$. The corrections $(I_k, I_\varepsilon) = (1, -)$ and $(I_k, I_\varepsilon) = (-, 1)$ refer to studies where independent sets of δ_k and δ_ε corrections were generated without their counterpart in the MK turbulence model.

results. It was verified that the trends observed in Fig. 3.6 are also present across all the other DNS cases.

Based on the results presented in Fig. 3.6, it was decided to study the effect of building independent sets of δ_k and δ_ε corrections. Employing a unique set of corrections can simplify the subsequent ML study, since the need to produce two-dimensional output pairs $(\delta_k, \delta_\varepsilon)$ is avoided. By applying the elbow method to calibrate the values of I_U for each set of independent predictions, it is found that $I_U = 100$ corresponds to a reasonable approximation as well. The individual corrections of δ_k and δ_ε can be found in Fig. 3.7 for all DNS cases, categorized in incompressible, high- and low-Mach number cases. Here, it must be noted that the maximum corrections for the low-Mach number cases are up to 4.5 times larger than the maximum of the corrections in the incompressible cases. Moreover, the different peaks and valleys found in the δ corrections for each DNS case present different shapes, relative magnitudes and even $Y^\star$ locations. For a few low-Mach number cases (right column), the δ_k distributions have values which are almost entirely positive.

While both δ_k and δ_ε corrections appear similar in Fig. 3.7, a detailed analysis revealed that the $\delta_\varepsilon/S_\varepsilon$ corrections contain gradients up to 83.5% higher than the maximum gradients observed for δ_k/S_k. Such sharper gradients would result in training a neural network which yields large changes in the predicted δ corrections based on smaller variations in the input features. Therefore, we decided to build a system based in δ_k/S_k corrections only.

3.5. MACHINE LEARNING PREDICTIONS

The final ML predictions were obtained by training the neural network architecture described in Fig. 3.4, and subsequently applying a weighted relaxation factor of $\alpha = 0.95$ in Eq. (3.27). This hyper-parameter was found to yield numerically stable CFD predictions for all cases, without modifying the δ corrections significantly. The variations in the error margins for every test case defined within the K-fold validation trials can be found in

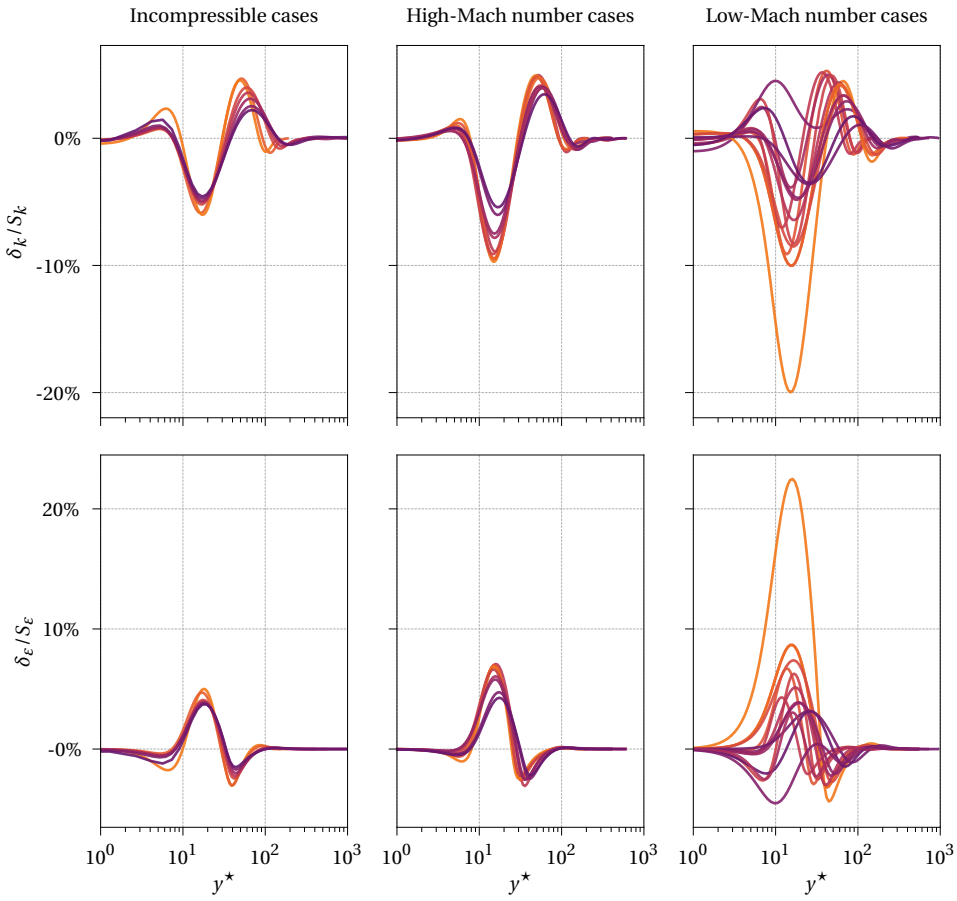


Figure 3.7: Independent set of field inversion corrections δ_k and δ_ε obtained for the MK turbulence model while employing $I_U = 100$.

Table 3.3. Here, the percentages for every validation case refer to the L-infinity norm of the differences between the velocity profiles for the baseline MK model (left) and the ML predictions (right) with respect to the DNS data. As can be seen, the ML predictions reduce the error margins in almost all CFD cases.

The best improvement margin can be found in the case $JFM.CRe_\tau^\star$ from the K-fold trial K-1, where the L-infinity norm of the errors was reduced from 23.4% to 4.0%. This result is important, since the DNS case $JFM.CRe_\tau^\star$ contains with the highest modelling errors with respect to the baseline MK model, and it requires the highest level of δ_k/S_k corrections according to Fig. 3.7. On the other hand, the worst deep learning predictions can be found in the case $IC.Re4200$ from the K-fold trial K-8, where the L-infinity norm is increased from 2.3% to 14.0%. This increase in error was expected, since the DNS case $IC.Re4200$ has the highest Reynolds number in our database: $Re_{\tau,w} = 4200$. The closest $Re_{\tau,w}$ value found in the remaining cases of the DNS database is $Re_{\tau,w} = 2000$, which is 2.1 times lower. Thus, the large error is simply the result of extrapolation. From a broader perspective, it can be concluded that our ML system is more accurate than the baseline MK model for the majority of the DNS cases in our database. In the few cases where deep learning performs slightly worse, the predictions are still reasonable.

A selection of the results for the K-fold trials with the best (K-1) and the worst (K-8) deep learning predictions can be found in Fig. 3.8. Here, the DNS cases shown contain the highest errors in the ML predictions for the velocity profiles within each K-fold trial. As can be observed in the sub-figures, the ML predictions (blue) for the velocity profiles are substantially closer to the DNS data (black) than the baseline MK model (red). The only exception in the sub-figures is the case $IC.Re4200$ within the K-fold trial (K-8), where the ML system was required to extrapolate as it was discussed before. Furthermore, Fig. 3.8 also presents the $\delta_{ML,k}$ corrections predicted by deep learning (blue), together with the reference $\delta_{FI,k}$ field inversion data (black). In most cases, the $\delta_{ML,k}$ corrections are qualitatively similar to $\delta_{FI,k}$, although significant differences can be observed at a given $y^\star$ location. However, these differences are small in magnitude, and the results indicate that they only produce minor changes in the velocity profiles.

The results for the DNS case $JFM.CRe_\tau^\star$ are analyzed in greater detail in Fig. 3.9. Here, a comparison is presented for the distribution in the errors of the velocity profiles between the baseline MK model and the ML predictions. The results for the ML predictions were sampled across all K-fold trials where the case $JFM.CRe_\tau^\star$ appeared as a validation case. The shaded area (gray) corresponds to the maximum and minimum bound of the ML errors observed across all the different K-fold trials. As can be observed, all the deep learning predictions are substantially more accurate than the baseline MK model. As it was discussed before, the $JFM.CRe_\tau^\star$ case is the most challenging. Therefore, the stability observed in the deep learning predictions for this DNS case shows that our ML architecture is able to achieve a robust behavior even in the presence of adverse modelling conditions.

The results of the non-dimensional feature importance ranking can be found in Fig. 3.10, which is determined using the integrated gradients (IG) method described in Section 3.3.4. The eddy viscosity μ_t/μ_w is the most important feature in the ranking. From a physical perspective, the eddy viscosity is the leading parameter that determines the diffusion of momentum, the turbulent kinetic energy and its dissipation. Other fea-

Table 3.3: Error margins for each test case in the different K-fold trials. The percentages under each case name indicate the error margins for the baseline MK turbulence model (left), and the deep learning predictions (right). All error percentages are calculated using the L-infinity norm for the difference between the velocity profiles of the models and the respective DNS data.

K-1	$JFM.CRe_\tau^\star$	Cv	$CRe_\tau^\star CPr^\star$	$M3.0R200$	$M4.0R200$	
	23.4% → 4.0%	7.8% → 2.9%	8.3% → 2.2%	9.4% → 5.2%	11.6% → 2.2%	
K-2	$JFM.CRe_\tau^\star$	$SRe_{\tau LL}^\star$	$GLCPr^\star$	$M1.7R200$	$M4.0R200$	$IC.Re950$
	23.4% → 6.3%	4.6% → 2.8%	6.6% → 1.3%	6.6% → 2.4%	11.6% → 5.4%	2.4% → 0.7%
K-3	$JFM.CRe_\tau^\star$	$JFM.GL$	$JFM.LL$	$M3.0R600$	$M4.0R200$	$IC.Re550$
	23.4% → 5.2%	9.6% → 4.8%	4.8% → 5.8%	10.0% → 5.9%	11.6% → 2.8%	2.4% → 1.0%
K-4	$JFM.CRe_\tau^\star$	$LL2$	$CP150$	$V\lambda SPr_{LL}^\star$	$M3.0R200$	$IC.Re950$
	23.4% → 10.2%	2.4% → 0.6%	2.9% → 1.1%	3.3% → 1.5%	9.4% → 7.4%	2.4% → 0.6%
K-5	$JFM.LL$	$LL1$	Cv	$CP395$	$M0.7R400$	$IC.Re550$
	4.8% → 3.4%	2.8% → 2.4%	7.8% → 3.2%	2.4% → 1.2%	3.2% → 1.4%	2.4% → 0.9%
K-6	$JFM.GL$	Cv	$CRe_\tau^\star CPr^\star$	$M0.7R400$	$M4.0R200$	$IC.Re2000$
	9.6% → 3.3%	7.8% → 4.1%	8.3% → 2.8%	3.2% → 1.5%	11.6% → 2.7%	2.4% → 2.5%
K-7	$JFM.CRe_\tau^\star$	GL	$CRe_\tau^\star CPr^\star$	$GLCPr^\star$	$M1.7R200$	$IC.Re550$
	23.4% → 4.2%	8.0% → 3.5%	8.3% → 1.5%	6.6% → 3.4%	6.6% → 2.3%	2.4% → 1.1%
K-8	$JFM.GL$	$SRe_{\tau GL}^\star$	$SRe_{\tau LL}^\star$	$V\lambda SPr_{LL}^\star$	$M1.7R400$	$IC.Re4200$
	9.6% → 1.0%	3.9% → 3.8%	4.6% → 3.8%	3.3% → 1.6%	4.7% → 1.6%	2.3% → 14.0%
K-9	$JFM.LL$	$GLCPr^\star$	$CP395$	$CP150$	$M1.7R600$	$IC.Re180$
	4.8% → 3.4%	6.6% → 1.4%	2.4% → 1.3%	2.9% → 2.2%	5.0% → 3.0%	3.0% → 1.6%
K-10	$JFM.CRe_\tau^\star$	$SRe_{\tau Cv}^\star$	$CRe_\tau^\star CPr^\star$	$CP395$	$M0.7R400$	$IC.Re950$
	23.4% → 8.6%	2.3% → 1.6%	8.3% → 2.5%	2.4% → 1.4%	3.2% → 2.2%	2.4% → 0.6%

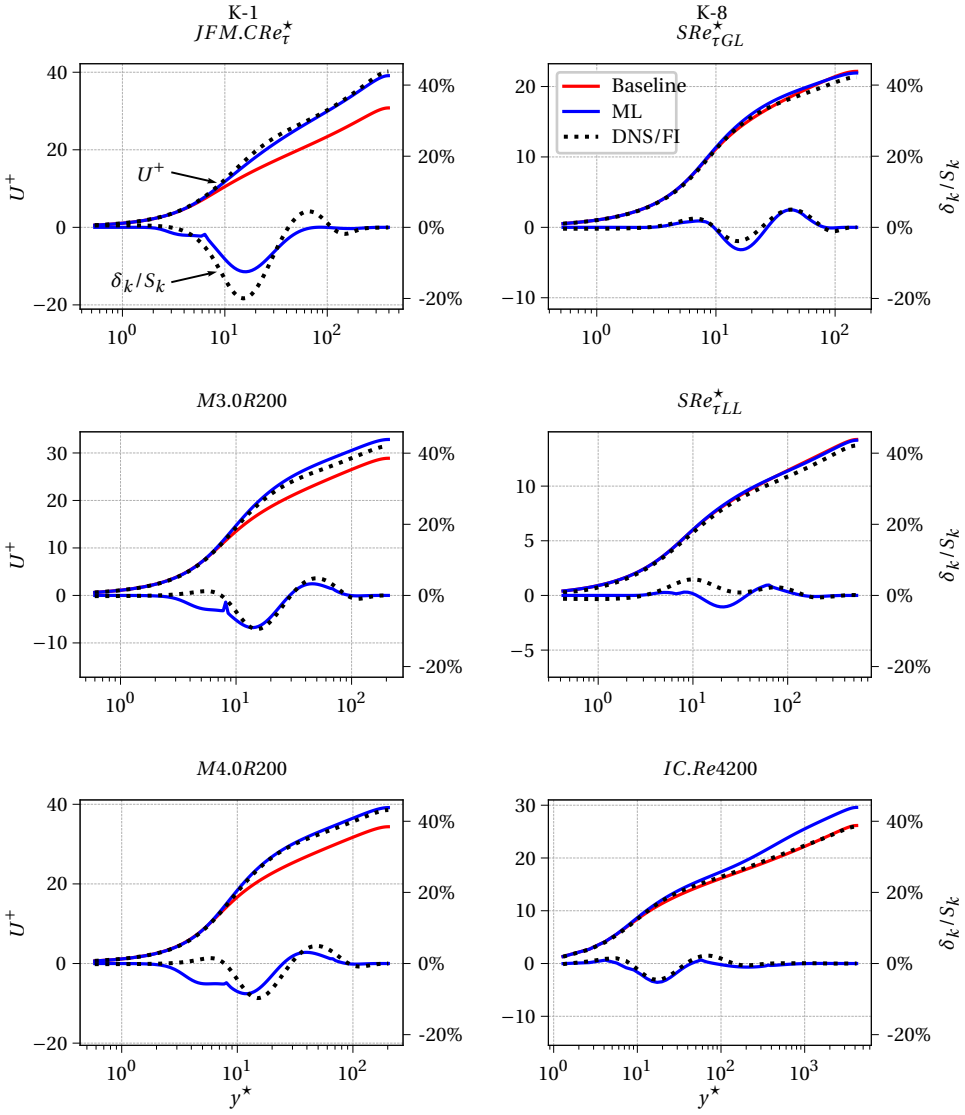


Figure 3.8: Selection of results at the extreme ends of the test error ranking for the K-fold validation trials after generating ML predictions for the independent sets of δ_k corrections required by the MK turbulence model. The upper U^+ curves represent the initial RANS velocity profiles (red lines), the deep learning velocity predictions (blue lines) and the reference DNS data (black dotted lines). The lower δ_k/S_k curves present the deep learning predictions after applying a weighted relaxation factor of 0.95 (blue lines) and the ground-truth labels for the field inversion values (black dotted lines).

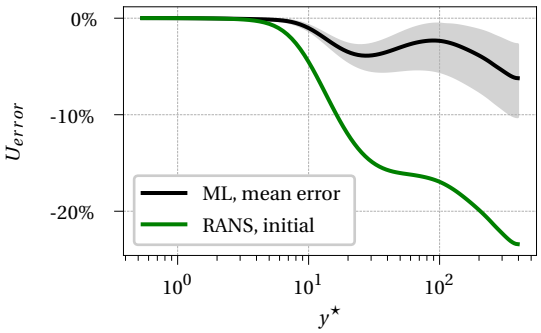


Figure 3.9: Results of the uncertainty quantification process followed for the DNS case $JFM.CRe_\tau^\star$ while employing the results of the K-fold validation runs K-1, K-2, K-3, K-4, K-7, K-10 after the prediction of the independent set of δ_k corrections for the MK turbulence model. The gray area corresponds to the maximum deviations observed in the neural network predictions across all K-fold trials.

tures, such as the turbulent production rate P_k/S_k, the turbulent kinetic energy k/M_k, the specific dissipation rate $\varepsilon/M_\varepsilon$ show less importance. On the other hand, the relatively low importance of the density and dynamic viscosity indicate that their variations are accounted for in $Y^\star$. This is in agreement with the modeling work performed by (Rodriguez et al., 2018).

3.6. CONCLUSIONS

In this chapter we used machine learning to improve the predictions of RANS turbulence modelling in channel flows subject to strong variations in their thermophysical properties. The methodology is based on a technique known as FIML proposed by (Parish & Duraisamy, 2016). In order to apply this method for our study, we have introduced several adaptations. For the field inversion methodology, we suggested a bold drive method with added momentum to drive the field inversion optimization proved to be stable and numerically efficient in over 450 optimization runs. As a result, this method can operate automatically requiring minimal attention from the user. The use of symbolic algebra solvers to generate expressions for the entries present in the matrices required

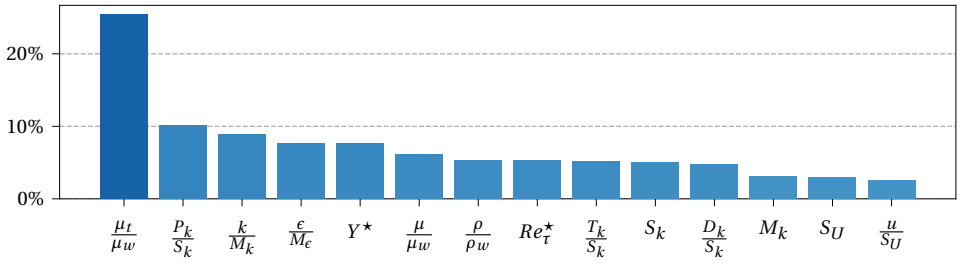


Figure 3.10: Feature importance ranking according to the integrated gradients method (Sundararajan et al., 2017).

by the discrete adjoint method in CFD is a valuable alternative, since the closed-form expressions generated are sparse-efficient. The overall shape of the corrections obtained can be controlled by employing cost functions containing adequate conversion terms (e.g., β vs. δ). Furthermore, L2 regularization helped to mitigate over-fitting and to reduce the importance of non-essential features.

Regarding the machine learning methodology, the use of an initial layer of logarithmic neurons followed by layers of hyperbolic tangent neurons resulted in a robust architecture, which was able to yield accurate predictions in nearly every case tested. By introducing a weighted relaxation factor methodology, the model was able to recover valuable trends from otherwise spurious predictions. It was demonstrated that our final deep learning predictions coupled with a CFD solver remained stable during all the cases tested. The overall behavior of the ML models indicates that the system is able to act as an excellent non-linear interpolator between DNS cases which are well-represented in the training set, and that the majority of the predictions for DNS cases sparsely represented in the dataset also show positive improvements. For the most challenging case, the baseline turbulence model produced an error of 23.4%, while the deep learning model displayed an average error of only 6.2%. Here, the error refers to the L-infinity norm of the difference between mean velocity of the model and the mean velocity of the DNS case. The case with the highest modelling errors only presented minor deviations in its velocity profile, and it corresponded to a case where the neural network was performing an extrapolation.

Finally, the importance of every feature in our system was ranked using the integrated gradients (IG) method. The IG method showed that the dimensionless eddy viscosity μ_t/μ_w corresponded to the most important feature, and that the semi-locally scaled wall distance $y^\star$ had greater importance than the individual values of μ/μ_w or ρ/ρ_w, since the variation in thermophysical properties is already accounted for in $y^\star$.

REFERENCES

Battiti, R. (1989). Accelerated backpropagation learning: Two optimization methods. *Complex Systems*, *3*(4).

Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, *52*(1), 477–508.

Chang, W., Chu, X., Binte Shaik Fareed, A. F., Pandey, S., Luo, J., Weigand, B., & Laurien, E. (2018). Heat transfer prediction of supercritical water with artificial neural networks. *Applied Thermal Engineering*, *131*, 815–824.

Coleman, G. N., Kim, J., & Moser, R. D. (1995). A numerical study of turbulent supersonic isothermal-wall channel flow. *J. Fluid Mech.*, *305*, 159–183.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314.

Durbin, P. A., & Reif, B. P. (2011). *Statistical theory and modeling for turbulent flows*. John Wiley & Sons.

He, S., Kim, W., & Bae, J. (2008). Assessment of performance of turbulence models in predicting supercritical pressure heat transfer in a vertical tube. *International Journal of Heat and Mass Transfer*, *51*(19), 4659–4675.

Hines, J. (1996). A logarithmic neural network architecture for unbounded non-linear function approximation. *Neural Networks, 1996., IEEE International Conference on*, *2*, 1245–1250 vol.2.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257.

Huang, P. G., Coleman, G. N., & Bradshaw, P. (1995). Compressible turbulent channel flows: DNS results and modelling. *J. Fluid Mech.*, *305*, 185–218.

Jiménez, J., & Hoyas, S. (2008). Turbulent fluctuations above the buffer layer of wall-bounded flows. *Journal of Fluid Mechanics*, *611*, 215–236.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations (ICLR)*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, *807*, 155–166.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., et al. (2017). Sympy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103.

Milano, M., & Koumoutsakos, P. (2002). Neural network modeling for near wall turbulent flow. *J. Comput. Phys.*, *182*(1), 1–26.

Mitliagkas, I., Zhang, C., Hadjis, S., & Ré, C. (2016). Asynchrony begets Momentum, with an Application to Deep Learning. *arXiv e-prints*, Article arXiv:1605.09774.

Morkovin, M. (1961). Effects of compressibility on turbulent flows. CNRS.

Mosteller, F., & Tukey, J. W. (1968). Data analysis, including statistics. In G. Lindzey & E. Aronson (Eds.), *Handbook of social psychology, vol. 2*. Addison-Wesley.

Myong, H., & Kasagi, N. (1990). A new approach to the improvement of k-ϵ; turbulence model for wall-bounded shear flows. *JSME international journal. Ser. 2, Fluids engineering, heat transfer, power, combustion, thermophysical properties*, *33*(1), 63–72.

Nemati, H., Patel, A., Boersma, B. J., & Pecnik, R. (2015). Mean statistics of a heated turbulent pipe flow at supercritical pressure. *International Journal of Heat and Mass Transfer*, *83*, 741–752.

Nie, Y., De Santis, L., Carratù, M., O'Nils, M., Sommella, P., & Lundgren, J. (2020). Deep melanoma classification with k-fold cross-validation for process optimization. *2020 IEEE International Symposium on Medical Measurements and Applications*, 1–6.

Parish, E., & Duraisamy, K. (2016). A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, *305*, 758–774.

Patel, A., Boersma, B. J., & Pecnik, R. (2016). The influence of near-wall density and viscosity gradients on turbulence in channel flows. *J. Fluid Mech.*, *809*, 793–820.

Patel, A., Peeters, J., Boersma, B., & Pecnik, R. (2015). Semi-local scaling and turbulence modulation in variable property turbulent channel flows. *Physics of Fluids*, *27*(9), 095101.

Patel, A. (2017). *Universal characterization of wall turbulence for fluids with strong property variations* [Dissertation]. Delft University of Technology.

Patel, A., Boersma, B. J., & Pecnik, R. (2017). Scalar statistics in variable property turbulent channel flows. *Phys. Rev. Fluids*, *2*, 084604.

Pecnik, R., & Patel, A. (2017). Scaling and modelling of turbulence in variable property channel flows. *Journal of Fluid Mechanics*, *823*, R1.

Pecnik, R., Rodriguez, G. J. O., Patel, A., & S., R. D. (2018). RANS channel. https://github.com/Fluid-Dynamics-Of-Energy-Systems-Team/RANS_Channel

Peeters, J. W. R., Pecnik, R., Rohde, M., van der Hagen, T. H. J. J., & Boersma, B. J. (2016). Turbulence attenuation in simultaneously heated and cooled annular flows at supercritical pressure. *Journal of Fluid Mechanics*, *799*, 505–540.

Peeters, J. (2022). On the effect of pseudo-condensation on the design and performance of supercritical co2 gas chillers. *International Journal of Heat and Mass Transfer*, *186*, 122441.

Rodriguez, G. J. O., Patel, A., S., R. D., & Pecnik, R. (2018). Turbulence modelling for flows with strong variations in thermo-physical properties. *International Journal of Heat and Fluid Flow*, *73*, 114–123.

Sanhueza, R. D., Smit, S. H., Peeters, J. W., & Pecnik, R. (2023). Machine learning for RANS turbulence modeling of variable property flows. *Computers & Fluids*, *255*, 105835. https://www.sciencedirect.com/science/article/pii/S0045793023000609

Singh, A. P., Duraisamy, K., & Zhang, Z. J. (2017, January). Augmentation of turbulence models using field inversion and machine learning. American Institute of Aeronautics; Astronautics.

Singh, A. P., Matai, R., Mishra, A., Duraisamy, K., & Durbin, P. A. (2017, June). Data-driven augmentation of turbulence models for adverse pressure gradient flows [0]. American Institute of Aeronautics; Astronautics.

Singh, A. P., Medida, S., & Duraisamy, K. (2017). Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, *55*(7), 2215–2227.

Smits, A. J., & Dussauge, J.-P. (2006). *Turbulent shear layers in supersonic flow*. Springer Science & Business Media.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 3319–3328.

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika*, *18*(4), 267–276.

Trettel, A., & Larsson, J. (2016). Mean velocity scaling for compressible wall turbulence with heat transfer. *Phys. Fluids*, *28*(2), 026102. https://doi.org/https://doi.org/10.1016/0045-7825(74)90029-2

Yoo, J. Y. (2013). The turbulent flows of supercritical fluids with heat transfer. *Annual Review of Fluid Mechanics*, *45*(1), 495–525.

4

NEURAL NETWORK FOR TURBULENT FLOWS PAST ROUGH SURFACES

Turbulent flows past rough surfaces can create substantial energy losses in engineering equipment. During the last decades, developing accurate correlations to predict the thermal and hydrodynamic behavior of rough surfaces has proven to be a difficult challenge. In this work, we investigate the applicability of convolutional neural networks to perform a direct image-to-image translation between the height map of a rough surface and its detailed local skin friction factors and Nusselt numbers. Additionally, we propose the usage of separable convolutional modules to reduce the total number of trainable parameters, and PReLU activation functions to increase the expressivity of the neural networks created. Our final predictions are improved by a new filtering methodology, which is able to combine the results of multiple neural networks while discarding non-physical oscillations likely caused by over-fitting. The main study is based on a new DNS database formed by 80 flow cases at a friction Reynolds number of $Re_\tau = 180$ obtained by applying random shifts to the Fourier spectrum of the grit-blasted surface originally scanned by (Busse et al., 2015). The results show that machine learning can accurately predict the skin friction values and Nusselt numbers for a rough surface. A detailed comparison with existing correlations in the literature revealed that the maximum errors generated by deep learning were only 8.1% for the global skin friction factors $\overline{C_f}$ and 2.9% for the Nusselt numbers $\overline{Nu}$, whereas the best classical correlations identified reached errors of 24.9% and 13.5% for $\overline{C_f}$ and $\overline{Nu}$ respectively. The deep learning results also proved stable with respect to rough surfaces with abrupt changes in their roughness elements, and only presented a minor sensitivity with respect to variations in the dataset size.

This chapter has been published in the International Journal of Heat and Fluid Flow, volume 103, p. 109204, 2023.

4.1. INTRODUCTION

Turbulent flows past rough surfaces can be found in a large variety of engineering applications. Irregular surfaces are often caused by external processes, such as bio-fouling, abrasion, machining, or corrosion. In most engineering applications, the presence of rough surfaces can substantially increase the drag resistance of transportation systems, and lower the efficiency of thermodynamic cycles. Despite these drawbacks, some rough surfaces under specific flow conditions can yield positive contributions. For example, certain rough surfaces can enhance heat transfer significantly, while having only a relatively small increase in pressure losses (Dipprey & Sabersky, 1963; Nilpueng & Wongwises, 2015; Ventola et al., 2014). Other surfaces were shown to reduce the total drag resistance of a turbulent flow (Golovin et al., 2016; Gong et al., 2021). One of the main challenges while working with rough surfaces is to predict the impact of a given surface topography on the drag resistance and heat transfer rates. Most correlations available in the literature predict the global skin friction factor $\overline{C_f}$ or the Stanton number $\overline{St}$ of a rough surface based on standard surface metrics, such as the root-mean-squared height variations, skewness, kurtosis, effective slope, forward-facing angles, or different auto-correlation functions (Jouybari et al., 2021; Napoli et al., 2008; Schultz & Flack, 2009; Thakkar et al., 2017). In this work, the operator $\overline{(...)}$ refers to the average across the whole surface. While traditional surface metrics offer a simplified framework to describe the geometry of rough surfaces, it has been shown in the literature that obtaining accurate predictions for the drag resistance of a generic rough surface remains an open challenge (Chung et al., 2021; Flack, 2018). The correlations found in the literature tend to be valid only for specific datasets, and the trends observed cannot be extrapolated to rough surfaces with widely varying characteristics. Moreover, several surface metrics, such as the skewness or the kurtosis, present mixed results across different studies (De Marchis et al., 2019; Flack & Schultz, 2010; Jouybari et al., 2021). As a result, a small group of surface features that produces a universal collapse of all known measurements for $\overline{C_f}$ or $\overline{St}$ with a low degree of dispersion has not been identified. One important reason behind the previous difficulties is that both the unique shape of roughness elements and their spatial organization affect turbulent flow fields differently (De Marchis et al., 2019; Forooghi et al., 2018; Yuan & Piomelli, 2014). The previous factors cannot be accounted for using traditional surface metrics, such as the skewness or the kurtosis, since these metrics do not encode information regarding the spatial organization of the roughness elements. An example of this issue can be found in Figure 4.1, where two irregular surfaces with identical roughness elements located at different positions are presented; both rough surfaces will have identical skewness or effective slope, but they will have a substantially different hydrodynamic behavior due to the alignment of the roughness elements (Forooghi et al., 2018).

Figure 4.1: Example of two rough surfaces with identical roughness elements and averaged local quantities, such as their skewness or their kurtosis, but different hydrodynamic behavior (Forooghi et al., 2018).

Machine learning may provide a robust alternative to traditional modelling for the

hydrodynamic behavior of rough surfaces. Machine learning for fluid mechanics has already proven useful in other applications, such as correcting existing RANS turbulence models (Parish & Duraisamy, 2016) or predicting the Reynolds stress tensor (Ling et al., 2016; Sandberg & Zhao, 2022; Weatheritt & Sandberg, 2016). In the context of rough surface modelling, (Jouybari et al., 2021) investigated the possibility to utilizing neural networks and Gaussian process regression to predict the equivalent Nikuradse sand-grain height k_s of rough surfaces based on traditional surfaces metrics. The parameter k_s corresponds to a length-scale that describes the hydrodynamic behavior of rough surfaces, and it can be used in combination with other correlations to predict the skin friction factor of a pipeline or channel flow (Moody, 1944; Nikuradse, 1933). However, correlations based on k_s can only predict the global behavior of a rough surface ($\overline{C_f}$ or $\overline{Nu}$), and thus they do not provide information regarding the local behavior of turbulence.

In this work, we investigate the applicability of machine learning systems based on convolutional neural networks to predict the local behaviour of turbulent flows past irregular surfaces. These systems present several advantages over traditional correlations, such as being able to process the input height map H(x,z) of a rough surface directly, and to perform predictions without requiring traditional surface metrics. Deep learning systems can perform predictions taking into account both the detailed shape of every roughness element, as well as their spatial distribution. Furthermore, deep learning systems can be modified to not only predict global quantities, such as $\overline{C_f}$ or $\overline{St}$, but also distributed local quantities. Additionally, ML systems can perform direct predictions for the thermal behavior of rough surfaces, without requiring previous estimations for C_f that could lead to additional uncertainty.

This chapter is organized as follows. In Section 4.2, the details of the DNS database are presented, including the methodology used to generate rough surfaces, the formulation of the CFD solver and all relevant physical parameters. Section 4.2.5 describes the numerical routine used to interpolate the wall forces and heat fluxes generated by a DNS solver using staggered grids and the immersed boundary method in CFD. In Section 4.3, the details of the machine learning systems developed are presented, as well as techniques used to improve the baseline architecture identified. In Section 4.4, the final results of the machine learning study are described, together with a comparison between the current results and traditional correlations found in the literature. In Section 4.5, the conclusions of the study are presented, along with future recommendations.

4.2. DNS DATABASE

4.2.1. METHODOLOGY FOR THE DNS SIMULATIONS

The methodology used to simulate turbulent flows past rough surfaces is based on the DNS framework described by (Peeters & Sandham, 2019). This formulation starts by considering a simple planar channel flow geometry to study the hydrodynamic behavior of rough surfaces. A schematic representation of the planar channel geometry with rough walls is shown in Figure 4.2. Here, the (x, y, z) coordinates correspond to the streamwise, wall-normal and spanwise directions respectively. Only one half of the computational domain is presented, since the upper section of the channel corresponds to a reflection of the bottom part with respect to the symmetry plane located at the

channel half-height $y = \delta$. The height function $H(x,z)$ shown in Figure 4.2 is defined such that $\int_{x,z} H(x,z)\, dx\, dz = 0$ across the entire domain. Periodic boundary conditions are considered in the $x-z$ directions in order to close the computational domain. The incompressible Navier-Stokes equations and the energy equation for fluids are solved in dimensionless form,

$$\nabla \cdot \mathbf{u} = 0, \tag{4.1}$$

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + Re_\tau^{-1} \nabla^2 \mathbf{u} + S_f, \tag{4.2}$$

$$\partial_t T + \mathbf{u} \cdot \nabla T = Pe_\tau^{-1} \nabla^2 T + S_q. \tag{4.3}$$

In eqs. (4.1-4.3), the variables $\mathbf{u}$, p, T correspond to the dimensionless velocity, the pressure and the temperature fields respectively. The parameter $Re_\tau = u_\tau \delta / \nu$ found in the momentum equations is the friction Reynolds number, which is defined using the average friction velocity u_τ at the rough walls, the channel half-height δ and the kinematic viscosity ν. The scaling factor $Pe_\tau = Re_\tau Pr$ corresponds to the friction Peclet number, where Pr is the molecular Prandtl number. The parameter S_f found in eq. (4.2) is a constant pressure-gradient term to force fluid motion, whereas S_q in eq. (4.3) is a constant volumetric heat source term to induce thermal gradients in the fluids. The usage of constant source terms in turbulent channel flows, such as S_f or S_q, has been extensively validated during the last decades (Busse et al., 2015; Kim et al., 1987; Orlandi & Leonardi, 2006; Peeters & Sandham, 2019; Thakkar et al., 2017). In order to account for the presence of rough surfaces $H(x,z)$ inside the computational domain, the DNS solver uses an implementation of the immersed boundary method based on the direct-forcing approach proposed by (Fadlun et al., 2000). In the first grid point inside the fluid domain, the velocity $\mathbf{u}$ and the temperature T are enforced by quadratic interpolation in the wall-normal direction. A detailed validation of this numerical method can be found in (Peeters & Sandham, 2019). While the DNS simulations are solved using dimensionless variables, it is important to note that the velocity and temperature fields are originally scaled by the friction velocity $u_\tau = \sqrt{\tau_w / \rho}$ and the friction temperature $T_\tau = q_w / (\rho c_p u_\tau)$. Here, the properties ρ and c_p refer to the fluid density and the specific heat capacity respectively, while the variables τ_w and q_w are the equivalent shear stresses and heat fluxes with respect to a smooth wall configuration.

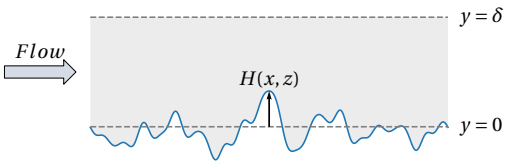


Figure 4.2: Schematic representation of the planar channel flow geometry considered for the DNS simulations. The variable $H(x,z)$ corresponds to the height function of the rough surfaces, which is defined based on the average plane at $y = 0$. The parameter δ is the half-channel height: $\delta = L_y/2$.

During all DNS simulations, it is assumed that the reference variables Pr, S_f and S_q are equal to unity. Due to global momentum and energy conservation, this further implies that the equivalent shear stresses τ_w and heat fluxes q_w also have unitary values: $\tau_w = q_w = 1$. Therefore, the present DNS simulations have well-defined momentum and energy balances. The goals of the DNS simulations are to quantify the variations in the bulk flow properties of turbulent flows, the changes in their boundary layer parameters, and the distribution of their local skin friction factors $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ across the rough surfaces. The formulas used to compute the values of $C_f(x,z)$ and $Nu(x,z)$ in Table 4.1 are the following,

$$C_f(x,z) = \frac{f_x(x,z)}{\frac{1}{2}\,\rho U_b^2}, \tag{4.4}$$

$$Nu(x,z) = \frac{q(x,z)}{\lambda\,(T_b - T_w)/L_y}. \tag{4.5}$$

In eqs. (4.4-4.5), the variables U_b and T_b correspond to the bulk velocity of the fluid and its bulk temperature respectively. The bulk Reynolds number is defined as $Re_b = U_b L_y/\nu$, where L_y corresponds to the channel size and $\nu = 1/Re_\tau$ is the kinematic viscosity considered for all DNS simulations. The parameters $\lambda = 1/(Re_\tau\ Pr)$ and $T_w = 0$ are the thermal conductivity for each fluid and the Dirichlet temperature boundary condition imposed at the rough walls respectively. The variables $f_x(x,z)$ and $q(x,z)$ are the local forces and heat fluxes per unit of area distributed across the rough surfaces.

As a reference, the local Stanton numbers $St(x,y)$ distributed across the rough surfaces are defined as,

$$St(x,z) = \frac{Nu(x,z)}{Re_b\ Pr}. \tag{4.6}$$

In principle, it is possible to reconstruct the thermal behavior of turbulent flows past rough surfaces by considering either the local Nusselt numbers $Nu(x,z)$ or Stanton numbers $St(x,z)$. However, deep learning systems designed to reconstruct the local Nusselt numbers $Nu(x,z)$ can predict the bulk temperature of a fluid T_b directly if the average heat flux $\overline{q}$ acting over the boundaries is known. This marks a large contrast with respect to deep learning systems designed to predict the local Stanton numbers $St(x,z)$, since these systems require explicit information regarding the bulk velocity of a fluid U_b before predicting its bulk temperature T_b. Therefore, predictions for T_b can only be obtained after coupling deep learning systems trained to reconstruct the local Stanton numbers $St(x,z)$ with an additional ML system to estimate U_b, which increases the uncertainty of the final predictions. Due to the previous reason, the current study focuses on predicting the local skin friction factors $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ for all the rough surfaces considered.

For each DNS case, the domain size has dimensions $(L_x \times L_y \times L_z) = (5.63 \times 2 \times 2.815)$ in the streamwise, vertical and spanwise directions respectively. The half-channel size is thus assumed to be equal to unity in all simulations: $\delta = L_y/2 = 1$. The reference length scale chosen for the rough surfaces (k) has a value ranging from $k/\delta = 16.12\%$ to 17.44% across all the rough surfaces considered for simulation. This length scale corresponds

to the mean-peak-to-valley height ($S_{z,5\times5}$) defined by (Thakkar et al., 2017). Based on this configuration, the grid size considered by (Peeters & Sandham, 2019) was employed during the current study, which has dimensions ($N_x \times N_y \times N_z$) = (280 × 280 × 140) for a friction Reynolds number of $Re_\tau = 180$. This grid size further ensures that approximately 12 grid points are used to discretize the smallest wavelength found in the definition of the height function $H(x,z)$ using a Fourier spectrum (Peeters & Sandham, 2019; Thakkar et al., 2017). The global statistics for the DNS simulations are presented later in Section 4.2.4, after the different types of rough surfaces considered in the study have been defined.

4.2.2. CATEGORIES OF ROUGH SURFACES

In order to create machine learning models, a DNS database was generated with 80 flow cases simulated with a friction Reynolds number of $Re_\tau = 180$. Numerical simulations are used instead of experimental measurements, since the machine learning formulation requires detailed information regarding the local skin friction factors $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ distributed across the rough surfaces. All the rough surfaces considered are generated by introducing phase shift variations ϕ_i to the Fourier spectrum of the grit-blasted originally surface scanned by (Busse et al., 2015). The height function $H(x,z)$ considered for the rough surfaces is the following,

$$H(x,z) = \sum_i R_i \cos\left(2\pi\left(x\,\frac{M_i}{L_x} + z\,\frac{N_i}{L_z}\right) - \phi_i\right). \tag{4.7}$$

In eq. (4.7), the terms R_i, M_i and N_i correspond to constants extracted from the Fourier spectrum defined by (Busse et al., 2015) in order to represent the grit-blasted surface scanned. Based on this formulation, the DNS database was generated by considering two categories of rough surfaces with a different methodology to define the phase shift component ϕ_i of every rough surface. The first group of rough surfaces, named Category I, was generated by considering 40 rough surfaces with random phase shift variations ϕ_i according to eq. (4.7). Despite the simplicity of the methodology used, the rough surfaces found in Category I contain different types of morphological features and clusters of roughness elements, as it can be observed in the left side of Figure 4.3. Therefore, predicting the local skin friction factors $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ generated by each rough surface is a challenging machine learning task, where it is necessary to predict the impact of each cluster of unique roughness elements in the local turbulent flow fields. The rough surfaces found in Category I are the main database used to train deep learning models during the current study.

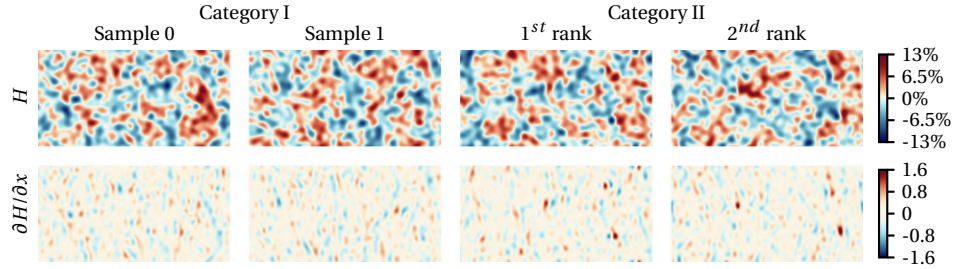


Figure 4.3: Examples of rough surfaces belonging to Categories I and II of the DNS database. All height percentages are scaled with respect to the half-channel height $\delta = L_y/2$. The magnitude of the gradients $\partial H/\partial x$ is based on the reference length scales defined for the DNS simulations. Each height map presented corresponds to the entire DNS domain, which is periodic in the streamwise and spanwise directions.

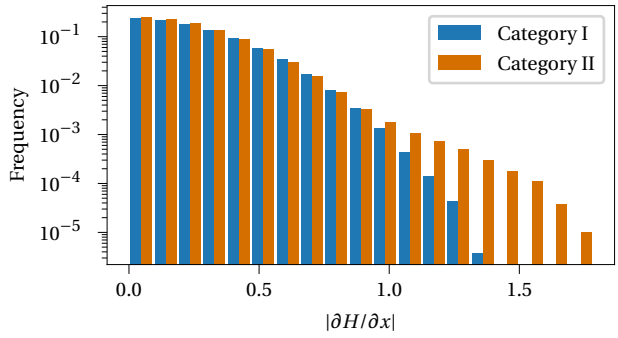


Figure 4.4: Distribution of local $|\partial H/\partial x|$ gradients for rough surfaces belonging to Categories I and II of the DNS database.

The second group of rough surfaces, named Category II, corresponds to a special selection of rough surfaces with unusually high slope angles in the streamwise direction $\alpha_x = tan^{-1}(\partial H/\partial x)$. These rough surfaces serve as a challenging validation scenario to test the robustness of the deep learning systems trained, since they contain roughness elements with abrupt changes in their shape, as well as substantial modifications in their local turbulent flow fields. The methodology used to generate these rough surfaces is presented in Section 4.2.3.

A graphical comparison between the height maps $H(x,z)$ and the gradients $\partial H/\partial x$ for the rough surfaces contained in Categories I and II can be found in Figure 4.3. The rough surfaces found in Category II contain regions with higher gradients than the examples shown from Category I. In Figure 4.4, a detailed histogram is presented regarding the distribution of $|\partial H/\partial x|$ gradients for all rough surfaces belonging to Categories I and II of the DNS database. According to the distributions shown, the rough surfaces from Category II contain significantly steeper slopes than any sample from Category I. Therefore, the rough surfaces from Category II correspond to outlier cases with large distortions in their roughness elements, which can be used to test the robustness of the deep learning system

to adverse conditions. Although it would be possible to include surfaces from Category II in the training sets, no significant improvements were observed in the validation performance during a preliminary study. Therefore, the surfaces from Category II are used exclusively to test the performance of the neural network for rough surfaces with features that fall outside Category I of the training sets.

4.2.3. GENERATION OF ROUGH SURFACES WITH HIGH SLOPE IN THE STREAMWISE DIRECTION

The rough surfaces found in Category II of the DNS database were generated by using a methodology specifically designed to detect rough surfaces with high slope in the streamwise direction $\partial H/\partial x$ across multiple regions of the DNS domain. The methodology starts by considering a collection of 1,000,000 rough surfaces, which contain purely random phase shift variations ϕ according to the definition of the height function $H(x,z)$ back in eq. (4.7). Then, all rough surfaces are ranked using an evaluation metric $\mathscr{M}$, which is defined using the highest values for $|\partial H/\partial x|$ found in every quadrant shown in Figure 4.5. The grey zones between the quadrants in Figure 4.5 correspond to regions where the values for $|\partial H/\partial x|$ are not evaluated. This helps the algorithm to avoid selecting rough surfaces where a single peak can represent the highest value for $|\partial H/\partial x|$ across multiple quadrants, since the idea is to obtain surfaces with multiple sharp roughness elements. The formula for the evaluation metric $\mathscr{M}$ is the following,

$$\mathscr{M} = \frac{\prod_i Q_i}{min(Q_i)}. \tag{4.8}$$

In eq. (4.8), the variables Q_i correspond to the largest values for $|\partial H/\partial x|$ found in every quadrant of Figure 4.5. The term $min(Q_i)$ found in the denominator of eq. (4.8) is used to create an evaluation metric, which effectively only depends on the three highest values registered for Q_i. This allows the algorithm to select rough surfaces that might also contain one quadrant with smooth roughness elements, and thus it helps to increase the diversity of the surfaces selected. Despite the previous fact, the product term $\prod_i Q_i$ found in eq. (4.8) implies that the remaining three quadrants of the surfaces must contain sharp roughness elements, or that at least one of the peaks found in the rough surfaces has very large $|\partial H/\partial x|$ values. Other formulas were also tested as a replacement for eq. (4.8), such as $\mathscr{M} = \sum_i Q_i - min(Q_i)$. However, the empirical results proved that many of top-ranked surfaces found by both methodologies corresponded to the same samples, and that the remaining surfaces had a similar morphology.

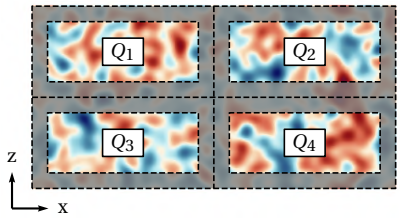


Figure 4.5: Quadrants used to evaluate the absolute value of $|\partial H/\partial x|$ while selecting rough surfaces belonging to category II of the DNS database. In the diagram shown, the horizontal and the vertical axis correspond to the streamwise and spanwise directions respectively.

4.2.4. BULK FLOW PROPERTIES AND BOUNDARY LAYER PARAMETERS FOR THE DNS DATABASE

In Table 4.1, the variations in the bulk flow properties and boundary layer parameters for the DNS simulations found in Categories I and II are presented. Here, it can be noted that all DNS cases for each category only present minor differences in the parameters being analyzed . However, the rough surfaces found in Category II present higher maximum values for all turbulent flow parameters, such as $\overline{C_f}$, $\overline{Nu}$, ΔU^+, ΔT^+. This implies that the distorted roughness elements found in these surfaces are able to produce minor changes in turbulence globally. The slightly lower values for the bulk Reynolds numbers Re_b found in rough surfaces from Category II can be explained since all DNS simulations are performed using a fixed momentum source term $S_f = 1$. This implies that rough surfaces which further enhance turbulence, and thus drag resistance, can reach the same target forces using slightly lower bulk velocities U_b, as it is observed in Table 4.1.

For more information, B.2 presents detailed sub-plots for each quantity reported in Table 4.1, as well as the values of different traditional surface metrics. Regarding these results, it must be noted that all the rough surfaces found in our dataset have a constant the root-mean-squared height of 0.0358. This implies that the phase shift components ϕ_i of the height function $H(x,z)$ do not influence its root-mean-squared value when integrated inside a periodic domain. However, other surface metrics contain substantially different values. For example, the skewness of the rough surfaces varies up to 662% with respect to its median value within the dataset.

The parameter $k^+_{s,eq}$ listed in Table 4.1 corresponds to the dimensionless Nikuradse sand-grain roughness height for the irregular surfaces considered. Please note that the flows in the current DNS simulations are in the transitionally rough regime. Obtaining the equivalent sand-grain roughness size k_s for all DNS cases would require many more simulations in order to find the fully-rough asymptote for each surface. Instead, to find $k^+_s = k_s u_\tau/\nu$, we compare our results with Nikuradse's equations (Nikuradse, 1933) directly, using the time-averaged velocity $U^+_c = U_c/u_\tau$ obtained from the DNS simulations at the channel center ($y = \delta$):

$$U^+_c - 5.75\,\log_{10}(\delta/k_s) = A(k^+_{s,eq}), \tag{4.9}$$

$$A(k^+_{s,eq}) = \begin{cases} 5.5 + 5.75\ \log_{10}(k^+_{s,eq}) & \text{if } k^+_{s,eq} < 3.55 \\ 6.59 + 3.5\ \log_{10}(k^+_{s,eq}) & \text{if } 3.55 \leq k^+_{s,eq} < 7.08 \\ 9.58 & \text{if } 7.08 \leq k^+_{s,eq} < 14.13 \\ 11.5 - 1.62\ \log_{10}(k^+_{s,eq}) & \text{if } 14.13 \leq k^+_{s,eq} < 67.61 \\ 8.48 & \text{if } 67.61 \leq k^+_{s,eq} \end{cases} \tag{4.10}$$

Eqs. (4.9-4.10) yield a 2.3% error when compared to the findings of (Thakkar et al., 2018) for a grit-blasted surface with $k^+_s = 26.1$.

Table 4.1: Variations in the bulk flow properties and boundary layer parameters for the DNS simulations present in Categories I and II. All variations are calculated with respect to the middle value for each range: $(max - min)/((min + max)/2)$.

4

Variable	Category I			Category II		
	Min.	Max.	Range	Min.	Max.	Range
$k^+_{s,eq}$	25.30	35.15	32.59 %	25.49	37.12	37.15 %
U_b	10.69	11.54	7.68 %	10.49	11.53	9.44 %
Re_b	3848	4156	7.68 %	3776	4150	9.44 %
$\overline{C_f}$	0.0150	0.0175	15.34 %	0.0150	0.0182	18.83 %
T_b	12.14	12.84	5.65 %	12.04	12.81	6.15 %
$\overline{Nu}$	28.02	29.66	5.65 %	28.12	29.9	6.15 %
$\overline{St}$	0.0068	0.0077	13.03 %	0.0068	0.0079	15.54 %
ΔU^+	4.09	5.07	21.46 %	4.14	5.26	23.91 %
ΔT^+	3.63	4.47	20.58 %	3.68	4.55	20.95 %

4.2.5. WALL FORCE AND HEAT FLUX INTERPOLATION

In order to calculate the local skin friction factors $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ acting over irregular surfaces, a post-processing routine was first developed to estimate the local forces **F** and heat fluxes Q distributed across the rough walls. This step is necessary, since the DNS solver uses a staggered grid to represent the velocity components u_x, u_y, u_z in the computational domain, and the immersed boundary method applied to CFD is used to account for the presence of the rough surfaces. Therefore, the local forces **F** and heat fluxes Q distributed across the rough surfaces are not readily available. The post-processing routine is mainly based on a Gauss integration scheme with face elements located over the rough walls. The integral equations are the following,

$$\mathbf{F} = \int_A \left(-P\,\mathbf{n} + \frac{1}{Re_\tau} \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) \cdot \mathbf{n} \right)\, dA, \tag{4.11}$$

$$Q = \frac{1}{Pe_\tau} \int_A \nabla T \cdot \mathbf{n}\ dA. \tag{4.12}$$

Here, A refers to the area of the rough surfaces in contact with the fluid, and $\mathbf{n}$ denotes the normal of the rough surfaces. The normal for the bottom side of the channel can be determined using,

$$\mathbf{n} = \left(\sqrt{1 + \frac{\partial H}{\partial x}^2 + \frac{\partial H}{\partial z}^2} \right)^{-1} \left[-\frac{\partial H}{\partial x} , 1 , -\frac{\partial H}{\partial z} \right]^T . \tag{4.13}$$

For the top side of the channel, this vector needs to be mirrored. The differential area dA for the face elements located over the rough surfaces is given by,

$$dA = \sqrt{1 + \frac{\partial H}{\partial x}^2 + \frac{\partial H}{\partial z}^2} \, dx \, dz. \tag{4.14}$$

When combining eqs. (4.11-4.14), the expression $\sqrt{1 + \partial H/\partial x^2 + \partial H/\partial z^2}$ cancels. Therefore, a Gauss-Legendre scheme with a sufficient number of integration points can be used to find the solution of the resulting polynomial integrals for the drag forces and heat fluxes.

In order to perform integration, the variables ($\mathbf{u}, P, T, H$) were reconstructed using shape functions based on a symmetric stencil of data points surrounding each face element. The degree of the polynomial terms chosen for every variable in each direction is listed in Table 4.2, whereas a graphical representation of original shape functions is given in Figure 4.6. Here, it can be noted that all interpolation schemes considered linear terms in the wall-normal direction (y), whereas mixed-order linear or quadratic terms were considered in the horizontal directions ($x - z$). The latter was necessary because the DNS solver employed a staggered discretization scheme to handle the velocity components $\mathbf{u} = (u_x, u_y, u_z)$. Therefore, the number of data points required to form a symmetric stencil was different in each direction. The usage of linear interpolation schemes in the wall-normal direction was validated empirically, since it was noted that quadratic terms yielded slightly higher errors with respect to both the global force and heat transfer balances. All integration areas considered for eqs. (4.11-4.12) were centered around the original $x - z$ coordinates for the pressure and temperature fields (P, T). The pressure field was extrapolated in the wall-normal direction (y) by considering the first two data points available above the rough surfaces. However, the velocity and temperature fields ($\mathbf{u}, T$) considered that the first layer of data points was located exactly over the rough surfaces at the corresponding $H(x, z)$ locations, with a value equal to the homogeneous Dirichlet boundary conditions ($\mathbf{u} = T = 0$).

Table 4.2: Order of the polynomial terms considered for the shape functions in every direction while interpolating the local forces and heat fluxes.

Variable	Streamwise (x)	Vertical (y)	Spanwise (z)
P	2	1	2
T	2	1	2
u_x	1	1	2
u_y	2	1	2
u_z	2	1	1
H	2	-	2

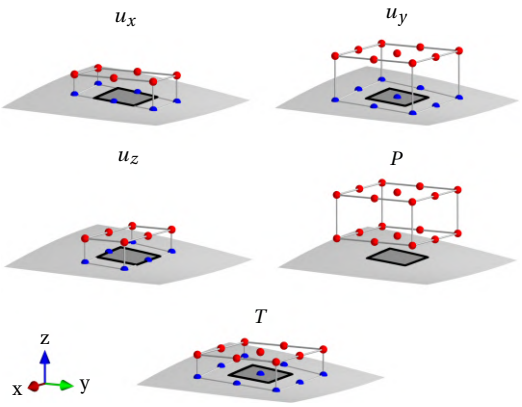


Figure 4.6: Schematic representation of the shape functions described in Table 4.2 for every flow variable considered. The red nodes correspond to data points extracted from the staggered grid of the DNS simulations, whereas the blue nodes are fixed points (with a value of zero) considered over the rough surfaces.

The numerical implementation of the current post-processing routine was written in PyTorch, since the entire procedure can be expressed as a sequence of parallelizable array operations. The process of gathering data points located in neighboring $x-z$ locations can be expressed using standard array shift operations due to the presence of periodic boundary conditions for the rough surfaces.

Finally, it is important to note that the local skin friction factors $C_f(x,z)$, Nusselt numbers $Nu(x,z)$ and Stanton numbers $St(x,z)$ for each face element can be computed using the formulas described back in eqs. (4.4-4.5). Within the context of these formulas, the local forces $f_x(x,z)$ and heat fluxes $q(x,z)$ per unit of area are given by,

$$f_x(x_i,z_j) = \frac{\mathbf{F}_{i,j}\cdot e_x}{\Delta x_i\,\Delta z_j}, \tag{4.15}$$

$$q(x_i,z_j) = \frac{Q_{i,j}}{\Delta x_i\,\Delta z_j}. \tag{4.16}$$

In eqs. (4.15-4.16), the sub-indexes (i,j) refer to the 2-D array position of each finite

element integration area considered. The variables $\mathbf{F}_{i,j}$ and $Q_{i,j}$ correspond to the results of the global integrals for the drag forces and heat transfer rates defined in eqs. (4.11-4.12) for each face element. The term e_x corresponds to the unitary vector in the x-direction, whereas the variables Δx_i and Δz_j refer to the size of each integration area in the $x-z$ directions. Regarding the denominator term ($\Delta x_i\ \Delta z_j$) found in eqs. (4.15-4.16), this expression corresponds to the projected area of each finite element with respect to the $x-z$ plane, or a smooth wall configuration. This definition is consistent with the definitions given in eqs. (4.4-4.5) for the skin friction values and Nusselt numbers, since in most engineering applications, the wetted area of a rough surface corresponds to an unknown quantity. Therefore, traditional definitions for the skin friction factor (C_f) or the Nusselt number (Nu) are given with respect to an equivalent smooth wall configuration.

4.3. MACHINE LEARNING

In general, the relationship between the input height map $H(x,z)$ of a rough surface and its local skin friction values $C_f(x,z)$ or Nusselt numbers $Nu(x,z)$ corresponds to a complex non-linear function. This Section presents a detailed description of the machine learning systems employed during the current study. First, in Section 4.3.1, a detailed overview of the neural networks created is given, together with a description of the training procedure and its cost function. Additionally, a novel smoothing procedure is introduced in Section 4.3.2.

4.3.1. NEURAL NETWORKS AND TRAINING PROCEDURE

During the current study, convolutional neural networks are employed since these models have been specifically designed to process images and to predict any target quantity of interest. The input height maps for rough surfaces $H(x,z)$ are processed as 2-D arrays formed by pixels, which correspond to grid points on the rough surface. These pixels are then translated into the corresponding local skin friction values $C_f(x,z)$ or Nusselt numbers $Nu(x,z)$ located at the center of each input image.

The deep learning systems employed during the study are formed by layers of depthwise separable convolution (DSC) modules (Chollet, 2017). The main benefits of this approach are that the total number of trainable parameters contained by convolutional neural networks is substantially reduced. Furthermore, an additional layer of non-linear activation functions is added between the depthwise and pointwise convolutional operators to increase the expressivity of the neural networks. For a convolutional operator with C internal channels and kernel size $K \times K$, the total number of parameters employed by DSC modules is reduced from C^2K^2 for a traditional kernel into only C^2+CK^2 parameters. Since $C \gg K$ inside the majority of deep learning systems, the previous analysis implies that the total number of trainable parameters is reduced by a factor of almost K^2.

The configuration of each DSC module employed during the study can be found in Figure 4.7. Here, it can be noted that traditional RELU activation functions have been replaced by PReLU operators (He et al., 2015). PReLU operators are able to rescale negative input values by a trainable parameter α, whereas RELU operators simply cancel negative values ($\alpha = 0$). As a result, PReLU operators can increase the expressivity of neural networks with a minimal computational cost, and they can facilitate the convergence

of the training procedure. Batch normalization is applied before each PReLU activation function to rescale the internal layers of the deep learning system and to train independent bias factors associated to every non-linear classifier. Furthermore, the usage of max-pooling operators was discarded during the current study. While max-pooling can extract physically relevant features generated by the internal layers of neural networks, we found during a preliminary study that neural networks with max-pooling or average-pooling operators did not yield higher accuracy. Due to this reason, we believe that max-pooling should only be considered as an alternative among other options while designing a neural network. In our final deep learning system, all reduction layers replaced max-pooling operators by additional convolutional modules.

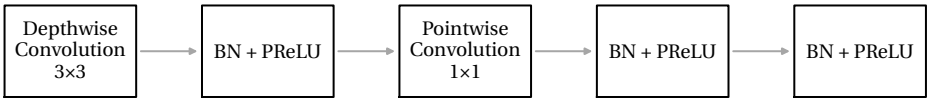


Figure 4.7: Schematic representation of the depthwise separable convolution (DSC) modules employed in the machine learning study. The abbreviation *BN* refers to the 2-D batch normalization operations applied before each PReLU activation function.

The global configuration of the deep learning architectures used to predict the local skin friction values $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ are listed in Tables 4.3 and 4.4 respectively. In these Tables, the columns C_{in}, C_{out}, K, D, AF refer to the number of input channels, output channels, kernel size, dilation and the presence of activation functions respectively. The increasing dilation levels found in Tables 4.3 and 4.4 correspond to a modification introduced into the neural networks to improve their time and space complexity by orders of magnitude, while still producing the same output as a classical convolutional neural network. A detailed explanation of this modification can be found in B.1. Figure 4.8 presents the total image size employed by the ML system described in Table 4.3 to predict $C_f(x,z)$ at a particular (x,z) location. As it can be observed, this ML system considers 0.48 $L_x \times 0.49\ L_z$ of the total domain size. Similarly, the ML system to predict $Nu(x,z)$ covers 0.91 $L_x \times 0.92\ L_z$ of the total domain. From a physical perspective, using the entire image size ($L_x \times L_z$) as input would be ideal. However, increasing the image size can lead to neural networks using an increased number of parameters, which in turn could result in over-fitting. Therefore, a trade-off must be considered between the input image size of a convolutional neural network and its number of trainable parameters. The architectures presented in Tables 4.3 and 4.4 were discovered by a simple neural architecture search algorithm based on greedy optimization (Lupo Pasini et al., 2021) for the leading ML configuration. Our algorithm aimed at optimizing the total number of layers, the number of internal convolutional channels, and the configuration of the dilation levels inside the neural networks for both $C_f(x,z)$ and $Nu(x,z)$. The smaller image sizes used to predict $C_f(x,z)$ are potentially caused by the fact that modelling this quantity is a more challenging task than predicting heat transfer, due to the non-locality of pressure drag effects. Therefore, the optimizer may have allocated a large number of trainable parameters to process small input images for $C_f(x,z)$, and it avoided using larger images to prevent over-fitting. For future reference, both of the deep learning architectures

described in Tables 4.3 and 4.4 have a total of 5,621 trainable parameters. While the greedy optimizer contained multiple options to increase the number of trainable parameters in each neural network separately, the final architectures chosen by the optimizer contained the same number of layers and trainable parameters.

It is important to highlight that our convolutional neural network always considers input images which are centered with respect to the prediction location. Therefore, the relative position of every roughness element is clearly defined in the streamwise and spanwise directions. Moreover, each pixel in the input images always represents the same physical size (in length units) within the DNS domain, and the row and column indexes of every 2-D image are aligned with the streamwise and spanwise directions respectively.

Table 4.3: Deep learning architecture to predict the local skin friction values $C_f(x,z)$. The parameters C_{in}, C_{out}, K and D refer to the number of input channels, output channels, kernel size, dilation and the presence of an activation function respectively.

Layer	C_{in}	C_{out}	K	D	AF
DSC	1	20	3	1×1	Yes
DSC	20	20	3	2×1	Yes
DSC	20	20	3	4×2	Yes
DSC	20	20	3	4×2	Yes
DSC	20	20	3	8×4	Yes
DSC	20	20	3	16×8	Yes
DSC	20	20	3	32×16	Yes
Conv. 1D	20	1	1	-	-

Table 4.4: Deep learning architecture to predict the local Nusselt numbers $Nu(x,z)$.

Layer	C_{in}	C_{out}	K	D	AF
DSC	1	20	3	1×1	Yes
DSC	20	20	3	2×1	Yes
DSC	20	20	3	4×2	Yes
DSC	20	20	3	8×4	Yes
DSC	20	20	3	16×8	Yes
DSC	20	20	3	32×16	Yes
DSC	20	20	3	64×32	Yes
Conv. 1D	20	1	1	-	-

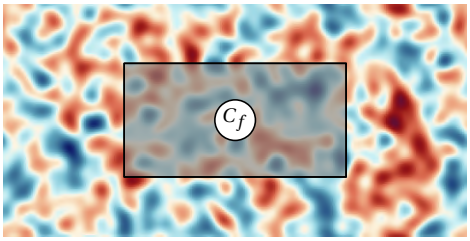


Figure 4.8: Physical image size (shaded region) employed by the deep learning architecture described in Table 4.3 to predict $C_f(x,z)$ at a particular (x,z) location (black circle). The shaded region must always be centered with respect to the predicted point, and thus it must be translated to obtain predictions for other (x,z) locations.

The training procedure for the deep learning systems was based on a L1 cost function instead of a traditional L2 penalization scheme,

$$\mathscr{J} = \sum_{k,i,j} \left| Y_{i,j}^{[k]\ dns} - Y_{i,j}^{[k]\ ml} \right|. \tag{4.17}$$

In eq. (4.17), the variables Y^{dns} and Y^{ml} correspond to the reference DNS data and the deep learning predictions respectively. Depending on the target of the study, the labels $Y_{i,j}$ can correspond to the local skin friction factors $C_f(x_i,z_j)$ or the local Nusselt numbers $Nu(x_i,z_j)$. The sub-index k refers to the identifier of the DNS case belonging to the training dataset that is being processed. The choice of a L1 cost function given by eq. (4.17) allows optimizers to train deep learning models which more accurately focus on minimizing the average errors for the local skin friction factors $C_f(x,z)$ or Nusselt numbers $Nu(x,z)$ than L2 loss functions, which tend to over-penalize small outlier regions. Additionally, during the training procedure, physics-informed data augmentation is performed to account for the fact that mirrored rough surfaces should have identical local skin friction values $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$. Moreover, since all rough surfaces are subject to periodic boundary conditions, the deep learning systems presented in Tables 4.3 and 4.4 utilize circular padding (Paszke et al., 2017) to account for the periodicity of the rough surfaces.

4.3.2. SMOOTHING PROCEDURE TO COMBINE THE PREDICTIONS OF MULTIPLE NEURAL NETWORKS

In order to avoid spurious oscillations in the final ML predictions, the results of the study were obtained by combining the results of two independently trained neural networks as shown in Figure 4.9. All neural networks were trained by splitting the DNS cases from Category I of the database into 32 training cases and 8 test cases. For each DNS case, only neural networks which included such case in their test set were considered. To explain the smoothing procedure, we consider two different sets of predictions, $Y^{[1]}$ and $Y^{[2]}$, generated by independent neural networks as shown in Figure 4.9. We imagine now that $Y^{[1]}$ and $Y^{[2]}$ can be combined such that a smoothed field β is obtained. In other words, β can be written as a weighted combination of multiple $Y^{[k]}$ fields,

$$\beta_{i,j} = \sum_k \alpha_{i,j}^{[k]}\, Y_{i,j}^{[k]}. \tag{4.18}$$

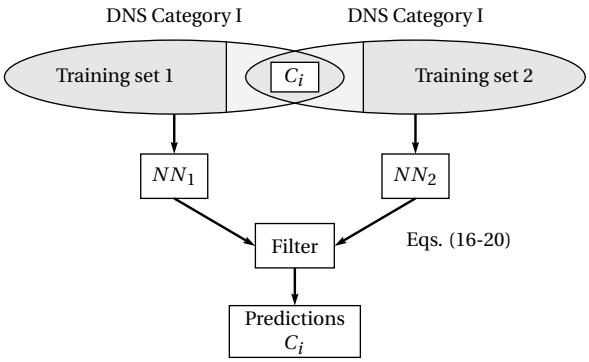


Figure 4.9: Schematic representation of the filtering procedure considering two neural networks. The variable C_i refers to the DNS case being predicted, which must belong to the test set of both neural networks. Only DNS cases from Category I of the database are included in the training sets. However, the case C_i may belong to Category I or II.

In eq. (4.18), the variable $Y_{i,j}^{[k]}$ represents the predictions of each individual neural network with sub-index $[k]$. The multipliers $\alpha_{i,j}^{[k]}$ represent the weight factors. These factors are subject to the following constraints,

$$0 \leq \alpha_{i,j}^{[k]} \leq 1, \tag{4.19}$$

$$\sum_k \alpha_{i,j}^{[k]} = 1. \tag{4.20}$$

Eqs. (4.19-4.20) imply that the value of each point in the combined field $\beta_{i,j}$ must be obtained though interpolation between the existing sets of predictions $Y_{i,j}^{[k]}$. To comply with these conditions, the values of the $\alpha_{i,j}^{[k]}$ coefficients were expressed using the softmax function,

$$\alpha_{i,j}^{[k]} = \frac{\exp s_{i,j}^{[k]}}{\sum_k \exp s_{i,j}^{[k]}}. \tag{4.21}$$

In eq. (4.21), the variable $s_{i,j}^{[k]}$ corresponds to an internal set of parameters employed by the softmax function. In order to create a smooth field $\beta_{i,j}$, the following cost function is used,

$$\begin{aligned} J_\beta &= \sum_{i,j} \left(\frac{\partial^2 \beta_{i,j}}{\partial x^2} \right)^2 + \left(\frac{\partial^2 \beta_{i,j}}{\partial z^2} \right)^2, \\ \beta_{i,j} &= \operatorname{argmin}(J_\beta). \end{aligned} \tag{4.22}$$

In eq. (4.22), the second-order derivatives of the target $\beta_{i,j}$ distribution are minimized by the cost function J_β. This formulation has favorable numerical properties, such as

being able to avoid isolated spikes in the target $\beta_{i,j}$ distributions, while still preserving physically meaningful peaks or gradients in the solutions generated by machine learning. All the ground-truth values for $C_f(x,z)$ and $Nu(x,z)$ in our DNS database contain smooth gradients, which implies that the previous formulation is well-suited to capture the trends observed in the physical data. Finally, the second-order expressions $\partial^2\beta_{i,j}/\partial x^2$ and $\partial^2\beta_{i,j}/\partial z^2$ in eq. (4.22) were approximated using central finite difference expressions,

$$\frac{\partial^2\beta_{i,j}}{\partial x^2} \approx \frac{\beta_{i+1,j} - 2\,\beta_{i,j} + \beta_{i-1,j}}{\Delta x_i^2}, \tag{4.23}$$

$$\frac{\partial^2\beta_{i,j}}{\partial z^2} \approx \frac{\beta_{i,j+1} - 2\,\beta_{i,j} + \beta_{i,j-1}}{\Delta z_j^2}. \tag{4.24}$$

The optimizer chosen to find $\beta_{i,j}$ corresponds to a gradient-descent scheme with adaptive learning rates (Battiti, 1989; Sanhueza, Smit, et al., 2023). This algorithm has several advantages, such as converging at optimal speeds using learning rates dynamically adjusted at run-time, or preventing divergence by performing line search until a smaller learning rate is found. From a practical perspective, another important property of the previous algorithm is that an early stopping criterion can be applied after a fixed number of iterations N_{max}, since the changes observed in $\beta_{i,j}$ decrease exponentially over time. An example of the results generated by the filtering methodology can be found in Figure 4.10. Here, it can be seen that spurious predictions are filtered by the methodology and that a smooth combined field is obtained as a result.

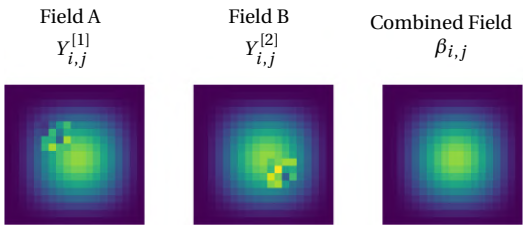


Figure 4.10: Synthetic example presenting the results of the filtering methodology described in Section 4.3.2.

Regarding the number of candidate predictions $Y^{[k]}$ passed to the algorithm, it is important to note that the current methodology is intended to work with only two prediction candidates $Y^{[1]}$ and $Y^{[2]}$. While more $Y^{[k]}$ fields could be considered in theory, the optimization algorithm will always try to choose the smoothest part of every input distribution. Therefore, using only two predictions candidates allows the algorithm to avoid numerical spikes in isolated regions, while still preserving physically meaningful gradients for $C_f(x,z)$ or $Nu(x,z)$.

4.4. RESULTS

4.4.1. LOCAL PREDICTIONS

The local skin friction factors $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ were computed by combining the machine learning systems and filtering methodology described in Section 4.3. The results for the local skin friction factors $C_f(x,z)$ can be found in Figure 4.11. Here, a comparison is presented between the machine learning predictions with the highest and the lowest accuracy with respect to the DNS data for rough surfaces belonging to Categories I and II of the DNS database. At it can be seen in Figure 4.11, the deep learning predictions show physically realistic trends, with the majority of the errors located in areas where extreme values of $C_f(x,z)$ are found. The latter is especially evident for the DNS cases with the highest prediction errors. Moreover, the machine learning predictions for rough surfaces belonging to Categories I and II of the DNS database show similar patterns, which implies that the deep learning system is able to handle irregular surfaces with larger local height gradients in the streamwise direction. However, it can also be noted that the deep learning system tends to under-predict negative $C_f(x,z)$ values in recirculation zones, such as the region indicated by the pairs of markers (A, A') or (B, B') in Figure 4.11. One possible reason for this behavior is that large negative $C_f(x,z)$ values rarely occur in our database, and thus the neural network can be biased towards predicting positive quantities.

The results of the deep learning predictions for the local Nusselt numbers $Nu(x,z)$ are presented in Figure 4.12. As it can be observed, there is a high degree of similarity between all machine learning predictions and the reference DNS data. Even for the worst case scenarios, the differences between the predicted $Nu(x,z)$ values and the DNS data are much smaller than the differences calculated for $C_f(x,z)$. However, it should be noted that the deep learning predictions for $Nu(x,z)$ tend to be more diffusive near regions with extreme values; see the regions indicated by the pairs of markers (A, A') or (B, B') in

Figure 4.12, for instance.

The results of the sensitivity analysis for the deep learning predictions with respect to the dataset size can be found in Figure 4.13. Here, histograms are presented comparing the distribution of local L1 errors for deep learning systems trained using 32, 16 and 8 rough surfaces. As it can be observed in the histograms, increasing the training set size from 8 to 16 or 32 rough surfaces has a positive effect on improving the accuracy of the momentum and heat transfer predictions. Moreover, it can be noted that predictions for the local Nusselt numbers $Nu(x,z)$ tend to have lower L1 errors than predictions for the local skin friction factors $C_f(x,z)$. One potential reason for the higher accuracy of the local Nusselt number $Nu(x,z)$ predictions is that heat transfer tends to occur mainly in peaks of the rough surfaces with high exposure to the incoming bulk flow, which can be easily identified by the deep learning system. In contrast, the local skin friction factors $C_f(x,z)$ are directly influenced by both pressure and viscous drag (Chung et al., 2021), which correspond to different physical processes. While viscous drag tends to occur close to the windward sides of local peaks, pressure drag can also occur in near stagnant regions due to the non-local pressure changes upstream or downstream with respect to large roughness elements. Therefore, predicting heat transfer may correspond to a simpler regression task than predicting momentum quantities.

It should also be noted that the magnitude of $C_f(x,z)$ can reach values up to 48.1 times higher than the average. This marks a large contrast with respect to the values for the local Nusselt numbers $Nu(x,z)$, which only reach magnitudes up to 14.0 times higher than the average in the datasets analyzed. Therefore, small relative differences between the deep learning predictions and the DNS data in regions where strong drag forces are expected will create much greater L1 errors than similar percentual differences in regions where large heat fluxes occur. The average error of the local ML predictions for $\overline{C_f}$ and $\overline{Nu}$ are 46.55% and 10.25% respectively. The worst 10% percentile of all local predictions have errors higher than 109.3% and 23.49% for $\overline{C_f}$ and $\overline{Nu}$ respectively. Note that these values are scaled with respect to the global values for $\overline{C_f}$ or $\overline{Nu}$ extracted from the DNS simulations. Due to this reason, the previous machine learning errors only correspond to a reference, and it is possible to reach values higher than 100% (of the average) in regions with large peaks for $C_f(x,z)$ or $Nu(x,z)$. For example, the peaks in the local skin frictions factors and Nusselt numbers can easily reach magnitudes of $C_f(x,z)/\overline{C_f} = 48.1$ and $Nu(x,z)/\overline{Nu} = 14.0$ within the DNS dabatase. Under these conditions, a machine learning prediction with a magnitude of $C_f(x,z)/\overline{C_f} \approx 43.29$ in the region with the highest recorded values for the local skin friction factors would only have a 10% local error, yet a 481% error with respect to the average value of $\overline{C_f}$. While the previous analysis may suggest that using local relative errors would be a better alternative, the opposite case can also occur. For instance, in a region with low $C_f(x,z)/\overline{C_f} \approx 0.001$ values, a ML prediction of $C_f(x,z)/\overline{C_f} \approx 0.01$ would be off by a factor of 10 locally, but only have a 0.9% error with respect to the global $\overline{C_f}$ value. Due to this reason, it was decided to always report the ML errors performing a comparison with respect to the average $\overline{C_f}$ values.

After a detailed analysis, it was found that the local slope angles of the rough surface and the errors in the machine learning predictions were only weakly correlated. However, the highest errors in the machine learning predictions are correlated with the presence of

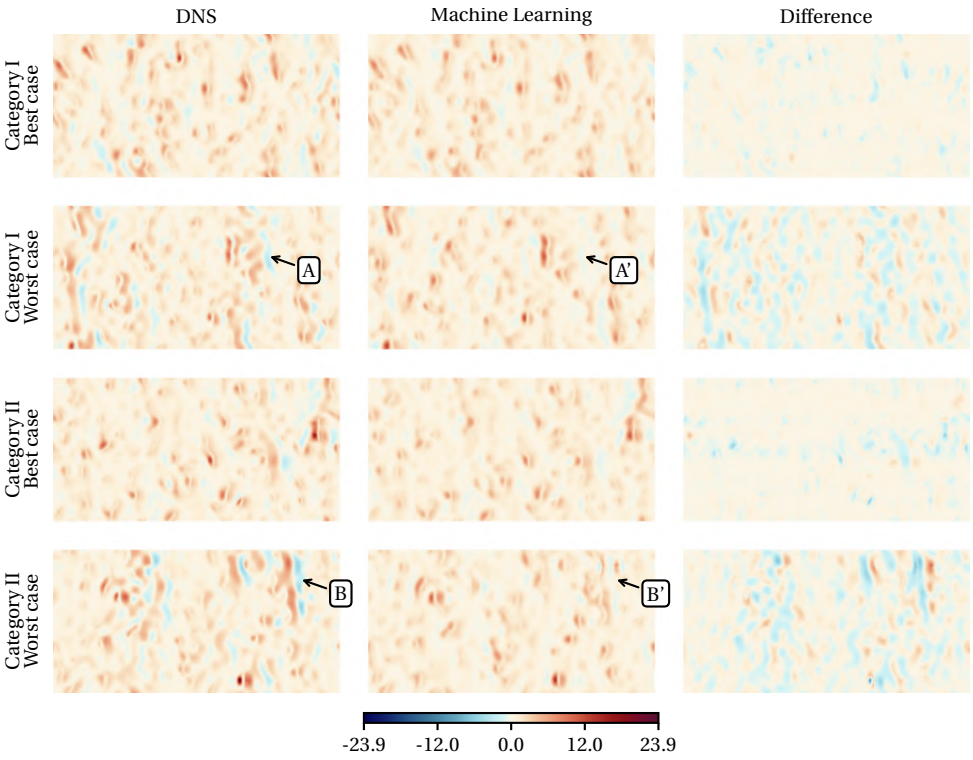


Figure 4.11: Comparison between the local skin friction factors $C_f(x,z)$ predicted by deep learning and the DNS data. The cases presented correspond to the highest and the lowest values for the L1-norm of the local errors. The values in the colormap indicate the ratio between $C_f(x,z)$ at every spatial location and the average skin friction factor $\overline{C_f}$ calculated according to the DNS data.

large clusters of roughness elements blocking the path of the incoming flow. The machine learning errors can occur either upstream or downstream with respect to these elements, due to pressure drag effects. This observation is not directly quantified by traditional metrics, but it can be clearly observed in examples, such as Figure 4.14. The cluster of roughness elements shown in this image creates multiple recirculation zones, which are not well-predicted by machine learning. Additionally, discrepancies can be observed at the top of the rough surfaces, and upstream of the cluster of roughness elements.

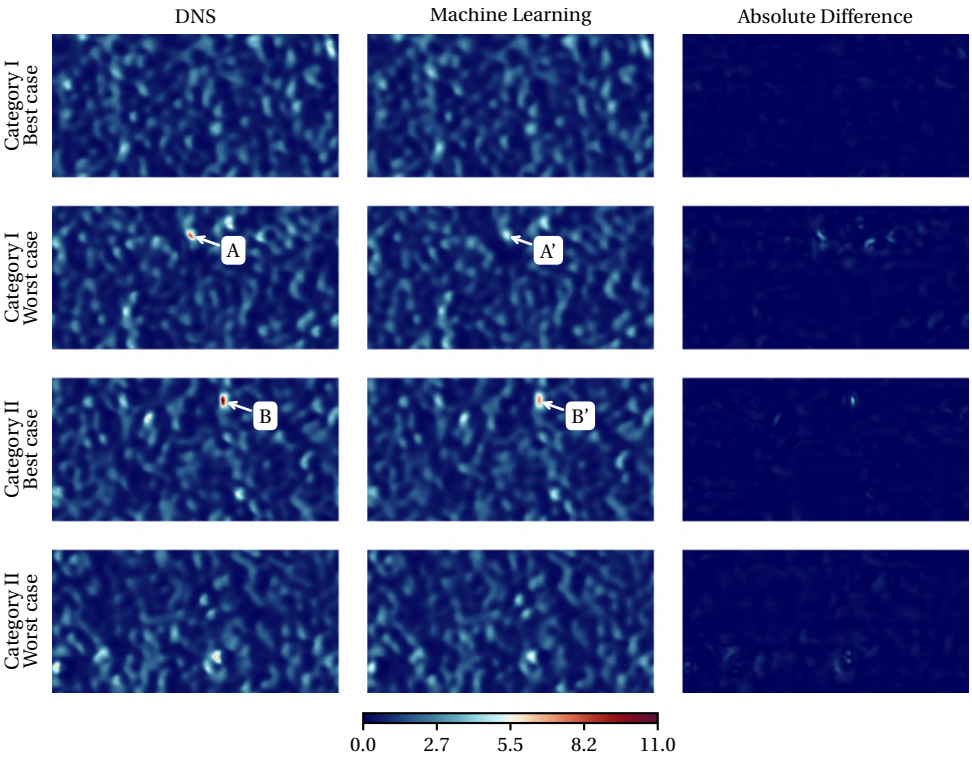


Figure 4.12: Comparison between the local Nusselt numbers $Nu(x,z)$ predicted using machine learning and the reference DNS data. The colormap indicates the ratio between the local values of the Nusselt numbers $Nu(x,z)$ and the average Nusselt numbers $\overline{Nu}$ given by the DNS data.

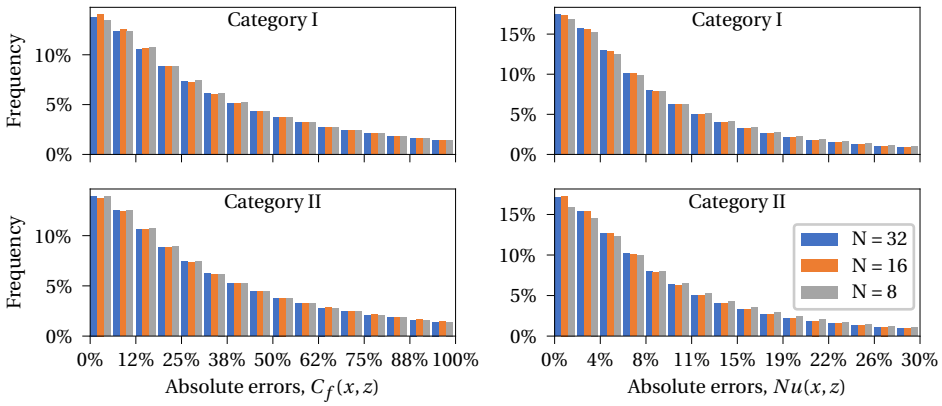


Figure 4.13: Changes in the distribution of the absolute errors for the local skin friction factors $C_f(x,z)$ and the Nusselt numbers $Nu(x,z)$ with respect to the training dataset size. All errors are scaled using the average DNS values for $\overline{C_f}$ or $\overline{Nu}$ within each case. The legend indicates the number of DNS cases used to train the deep learning models, which were extracted from Category I of the database. The title of each subplot indicates to which category the test cases plotted in each histogram belong. For the DNS cases from Category I of the database, multiple deep learning models were trained, such that each sample could be considered as a held-back test case in a separate study.

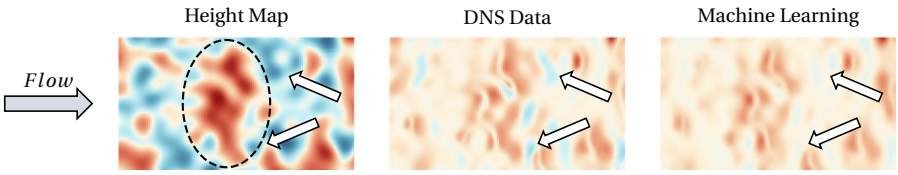


Figure 4.14: Example of the correlation between large clusters of roughness elements blocking the incoming flow and large errors in the machine learning predictions distributed nearby.

4.4.2. GLOBAL PREDICTIONS

A study was conducted to analyze the errors observed for the global skin friction factors ($\overline{C_f}$) and Nusselt numbers ($\overline{Nu}$) for every rough surface. The results were compared with traditional correlations found in the literature, which are based on the dimensionless Nikuradse sand-grain roughness height $k^+_{s,eq}$. After performing a preliminary analysis, it was determined that the correlation developed by (Flack & Schultz, 2010) yielded the most accurate predictions,

$$k_{s,eq} = 4.43\ S_q\ (1 + S_{sk})^{1.37}. \tag{4.25}$$

In eq. (4.25), the variables S_q and S_{sk} are the root-mean-squared height variations of the rough surface and its skewness, respectively (Thakkar et al., 2017),

$$S_q = \sqrt{\frac{1}{n_x n_z} \sum_{i,j}^{n_x,n_z} H^2_{i,j}}, \tag{4.26}$$

$$S_{sk} = S_q^{-3} \frac{1}{n_x n_z} \sum_{i,j}^{n_x,n_z} H^3_{i,j}. \tag{4.27}$$

In eqs. (4.26-(4.27), the variable $H_{i,j}$ corresponds to a 2-D array with the height of the rough surfaces at discrete $[i, j]$ locations in the $x-z$ directions respectively. After obtaining the Nikuradse sand-grain roughness height $k_{s,eq}$, the value of $k^+_{s,eq}$ can be computed from:

$$k^+_{s,eq} = k_{s,eq} \frac{u_\tau}{\nu}. \tag{4.28}$$

In order to obtain the skin friction factor $\overline{C_f}$ associated with $k^+_{s,eq}$ in the Nikuradse diagram (Nikuradse, 1933), the following formulas can be used (Peeters & Sandham, 2019; Thakkar et al., 2017):

$$\overline{C_f} = \frac{2}{\left(U^+_b\right)^2}, \tag{4.29}$$

$$U^+_b = U^+_{b,s} - \Delta U^+, \tag{4.30}$$

$$\Delta U^+ = U^+_{c,s} - U^+_c. \tag{4.31}$$

In eqs. (4.29-4.31), the variable U_b^+ corresponds to the dimensionless bulk velocity. The parameters $U_{c,s}^+$ and $U_{b,s}^+$ correspond to the dimensionless time-averaged velocity of a turbulent channel flow with smooth walls at the channel center ($y = \delta$) and its bulk velocity respectively. Based on existing DNS data (Peeters & Sandham, 2019), these variables were assigned values of $U_{c,s}^+ = 18.64$ and $U_{b,s}^+ = 15.79$ for a turbulent channel flow at $Re_\tau = 180$. The dimensionless velocity at the channel center (U_c^+) can be calculated using eqs. (4.9-4.10) from Section 4.2.4 based on the value of $k_{s,eq}^+$ in eq. (4.28). The Nusselt number for the rough surfaces is then calculated from the correlation for the Stanton number given by (Dipprey & Sabersky, 1963),

$$\overline{St} = \frac{C_f/2}{1 + \sqrt{C_f/2}\left\{k_f\left[Re_b\sqrt{C_f/2}\,(\epsilon_s/D)\right]^{0.2} Pr^{0.44} - 8.48\right\}} \tag{4.32}$$

In eq. (4.32), the parameters k_f has a value of 5.19. The ratio ϵ_s/D found in eq. (4.32) is given by,

$$\frac{\epsilon_s}{D} = \exp\left(\frac{3 - 1/\sqrt{C_f/2}}{2.5}\right) \tag{4.33}$$

The Nusselt number $\overline{Nu}$ associated to the correlation of (Dipprey & Sabersky, 1963) is then calculated from the Stanton number using the formula $\overline{Nu} = \overline{St}\; Re_b\; Pr$ given by eq. (4.6). The bulk Reynolds number found in eq. (4.32) can be calculated from the dimensionless bulk velocity U_b^+ given by eq. (4.29), since $u_\tau = 1$ in all DNS cases.

The results of the study regarding the accuracy of the global skin friction factors $\overline{C_f}$ and Nusselt number predictions $\overline{Nu}$ can be found in Figure 4.15. As it can be observed, the deep learning system is substantially more accurate than traditional correlations while predicting both momentum $\overline{C_f}$ and heat transfer $\overline{Nu}$ parameters. The maximum errors for the skin friction coefficients $\overline{C_f}$ in the deep learning system reached values of 8.07%, whereas the system with traditional correlations reached errors up to 24.9%. In the case of heat transfer, the maximum errors for the local Nusselt numbers were 2.87% and 13.5% for the deep learning system and the traditional correlations respectively. Both of these results are satisfactory for machine learning. Moreover, only small differences were found between the accuracy of the deep learning system while making predictions for rough surfaces belonging to both Categories I and II of the DNS database.

Additionally, it is important to analyze the ratio between the errors in the machine learning predictions and the variations observed in the DNS database for $\overline{C_f}$ and $\overline{Nu}$. If the machine learning errors have an absolute value smaller than the variations observed for $\overline{C_f}$ and $\overline{Nu}$ in the DNS data, it means that machine learning is accurately predicting the physical changes observed. Given this context, the results of Table 4.1 indicate that the maximum variations observed in the DNS data for $\overline{C_f}$ and $\overline{Nu}$ are 18.8% and 6.5% respectively. Therefore, the maximum errors found in the machine learning predictions of 8.1% for $\overline{C_f}$ and 2.9% for $\overline{Nu}$ can be regarded as small. The ratio between the maximum variations observed in the DNS data and the deep learning predictions is 2.3 for both $\overline{C_f}$ (18.8%/8.1%) and $\overline{Nu}$ (6.5%/2.87%). Therefore, it can be concluded that deep learning

constitutes a valid alternative to generate improved predictions for flow parameters, such as $\overline{C_f}$ or $\overline{Nu}$, if enough training data is collected.

To further put the neural network predictions into perspective, the maximum and the mean relative errors are compared with other predictive methods in Table 4.5, namely, the correlations of (Flack & Schultz, 2010) and (Dipprey & Sabersky, 1963), the re-calibrated versions of the latter, and simply considering the average values found in the training sets. The last method refers to averaging the $n_s - 1$ samples found in the training set of a predictive model for a target rough surface. The re-calibrated version of the correlation by (Flack & Schultz, 2010) has the following form: $k_{s,eq} = a\, S_q\, (1 + S_{sk})^b$, where a varies from 4.114 to 4.129 in every trained model, and b varies from 0.272 and 0.322. Additionally, in order to analyze heat transfer, the correlation from (Dipprey & Sabersky, 1963) was also re-calibrated to fit our training data, the parameterized version of the correlation has the form:

$$\overline{St} = \frac{C_f/2}{1 + \sqrt{C_f/2}\left\{a\left[Re_b\sqrt{C_f/2}\,(\epsilon_s/D)\right]^b Pr^{0.44} - c\right\}}, \tag{4.34}$$

$$\frac{\epsilon_s}{D} = \exp\left(\frac{d - 1/\sqrt{C_f/2}}{e}\right). \tag{4.35}$$

In eqs. (4.34-4.35), the re-calibrated parameters have the following values after training: $a \in [5.213, 5.222]$, $b \in [0.230, 0.234]$, $c \in [8.453, 8.461]$, $d \in [3.011, 3.016]$ and $e \in [2.562, 2.589]$. The mean relative errors are calculated as follows:

$$L_{1\,\overline{C_f}} = \frac{1}{n_s}\sum_{k=1}^{n_s}\left|\frac{\overline{C_f}^{[k]\ pred} - \overline{C_f}^{[k]\ dns}}{\overline{C_f}^{[k]\ dns}}\right| \tag{4.36}$$

$$L_{1\,\overline{Nu}} = \frac{1}{n_s}\sum_{k=1}^{n_s}\left|\frac{\overline{Nu}^{[k]\ pred} - \overline{Nu}^{[k]\ dns}}{\overline{Nu}^{[k]\ dns}}\right| \tag{4.37}$$

In eqs. (4.36-4.37), n_s refers to the number of samples in our dataset. As it can be observed in Table 4.5, our machine learning model significantly outperforms the traditional correlations from (Flack & Schultz, 2010) and (Dipprey & Sabersky, 1963). However, the machine learning model for $\overline{C_f}$ only displays a marginal improvement compared to the predictions of the re-calibrated correlation and the $n_s - 1$ averaged values. This result stems from the fact that the machine learning model was not trained to predict $\overline{C_f}$ and $\overline{Nu}$ directly, but it was rather trained to predict the local $C_f(x,z)$ and $Nu(x,z)$ values. Therefore, the deep learning predictions for $\overline{C_f}$ and $\overline{Nu}$ reported in Table 4.5 correspond to an emergent quantity, rather than an objective.

Table 4.5: Comparison of the maximum and the mean relative errors in the predictions for the global skin friction factors $\overline{C_f}$ and Nusselt numbers $\overline{Nu}$ of the rough surfaces using different models for rough surfaces belonging to Categories I and II of the DNS database.

	$L_{1\,\overline{C_f}}$	$L_{\infty\,\overline{C_f}}$	$L_{1\,\overline{Nu}}$	$L_{\infty\,\overline{Nu}}$
Machine learning	2.69%	8.07%	0.94%	2.87%
Correlation by (Dipprey & Sabersky, 1963; Flack & Schultz, 2010)	7.89%	24.88%	10.05%	13.49%
Re-calibrated correlations from (Dipprey & Sabersky, 1963; Flack & Schultz, 2010)	2.78%	8.92%	1.02%	3.74%
$n_s - 1$ averaged values	2.95%	11.12%	1.03%	3.86%

Regarding the generalizability of the model, we believe that the current convolutional neural network requires training data with similar flow characteristics, or boundary conditions, as the target test cases to make accurate predictions. In order to utilize different Reynolds numbers Re_τ or Prandtl numbers Pr, further research is required to establish the best methodology to include these quantities as input to the neural network.

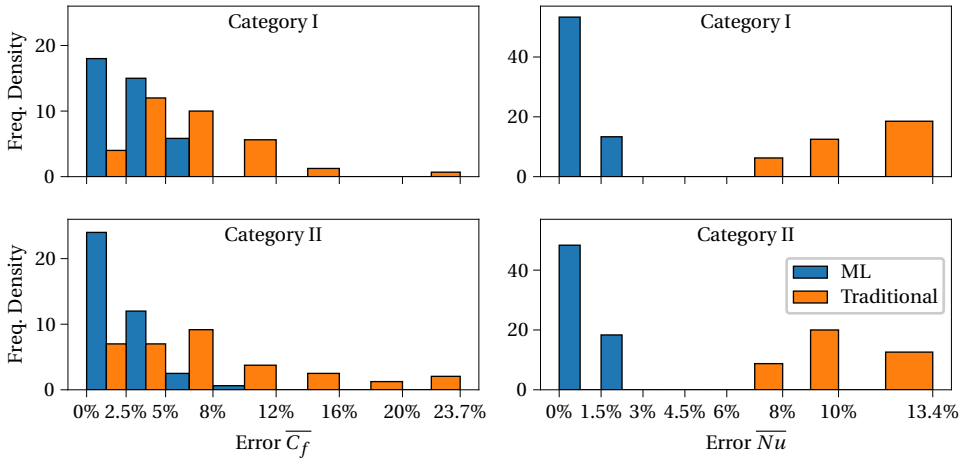


Figure 4.15: Histograms comparing the distribution of errors for the skin friction values $\overline{C_f}$ and the Nusselt numbers $\overline{Nu}$ between the deep learning model and the traditional correlations of (Flack & Schultz, 2010) and (Dipprey & Sabersky, 1963). The title of each subplot indicates the category of the DNS database to which the test cases found in each histogram belong. For rough surfaces belonging to Category I of the DNS database, the mean relative errors in the machine learning predictions are 2.92% for $\overline{C_f}$ and 0.90% for $\overline{Nu}$. Similarly, for Category II, the mean relative errors are 2.46% and 0.97% for $\overline{C_f}$ and $\overline{Nu}$ respectively.

4.5. CONCLUSIONS

This study presented a deep learning architecture capable of predicting detailed maps for the local skin friction factors $C_f(x,z)$ and Nusselt numbers $Nu(x,z)$ of irregular surfaces.

The results show that machine learning is able to achieve reliable results while predicting both global force and heat transfer parameters. A sensitivity study with respect to the dataset size also revealed a significant reduction in the errors for the momentum and heat transfer predictions once the dataset size is increased from 8 to 32 DNS cases. The comparisons performed with respect to traditional correlations proved that deep learning is a valid alternative to generate improved predictions for important flow parameters, such as the skin friction factors $\overline{C_f}$ or the Nusselt numbers $\overline{Nu}$. Moreover, the machine learning systems trained were able to obtain reliable predictions while working with rough surfaces containing abrupt changes in their roughness elements. The maximum error for $\overline{C_f}$ using traditional correlations was 24.9%, whereas deep learning only reached errors of 8.1%. Similarly, the maximum errors for $\overline{Nu}$ were reduced from 13.5% using traditional correlations to only 2.9% using deep learning. Regarding the local errors for $C_f(x,z)$ and $Nu(x,z)$, the probability density functions revealed that the median of the errors were lower than 28.42% and 6.37% respectively. The reason for the higher errors observed in C_f is likely related to non-local effects caused by pressure drag, as there is no analogous phenomenon for heat transfer. Therefore, it is recommended to perform further research regarding the creation deep learning models to predict the behavior of turbulent flows past rough surfaces.

REFERENCES

Battiti, R. (1989). Accelerated backpropagation learning: Two optimization methods. *Complex Systems, 3*(4).

Busse, A., Lützner, M., & Sandham, N. D. (2015). Direct numerical simulation of turbulent flow over a rough surface based on a surface scan. *Computers & Fluids, 116*, 129–147. https://doi.org/https://doi.org/10.1016/j.compfluid.2015.04.008

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *IEEE Conference on Computer Vision and Pattern Recognition*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195

Chung, D., Hutchins, N., Schultz, M. P., & Flack, K. A. (2021). Predicting the drag of rough surfaces. *Annual Review of Fluid Mechanics, 53*(1), 439–471. https://doi.org/10.1146/annurev-fluid-062520-115127

De Marchis, M., Milici, B., & Napoli, E. (2019). Large eddy simulations on the effect of the irregular roughness shape on turbulent channel flows. *International Journal of Heat and Fluid Flow, 80*, 108494. https://doi.org/https://doi.org/10.1016/j.ijheatfluidflow.2019.108494

Dipprey, D., & Sabersky, R. (1963). Heat and momentum transfer in smooth and rough tubes at various prandtl numbers. *International Journal of Heat and Mass Transfer, 6*(5), 329–353. https://doi.org/https://doi.org/10.1016/0017-9310(63)90097-8

Fadlun, E., Verzicco, R., Orlandi, P., & Mohd-Yusof, J. (2000). Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics, 161*(1), 35–60. https://doi.org/https://doi.org/10.1006/jcph.2000.6484

Flack, K. A. (2018). Moving beyond moody. *Journal of Fluid Mechanics*, *842*, 1–4. https://doi.org/10.1017/jfm.2018.154

Flack, K. A., & Schultz, M. P. (2010). Review of Hydraulic Roughness Scales in the Fully Rough Regime [041203]. *Journal of Fluids Engineering*, *132*(4). https://doi.org/10.1115/1.4001492

Forooghi, P., Stripf, M., & Frohnapfel, B. (2018). A systematic study of turbulent heat transfer over rough walls. *International Journal of Heat and Mass Transfer*, *127*, 1157–1168. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2018.08.013

Golovin, K. B., Gose, J. W., Perlin, M., Ceccio, S. L., & Tuteja, A. (2016). Bioinspired surfaces for turbulent drag reduction. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2073), 20160189. https://doi.org/10.1098/rsta.2016.0189

Gong, W., Shen, J., Dai, W., Li, K., & Gong, M. (2021). Research and applications of drag reduction in thermal equipment: A review. *International Journal of Heat and Mass Transfer*, *172*, 121152. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2021.121152

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision*, 1026–1034. https://doi.org/10.1109/ICCV.2015.123

Jouybari, M. A., Yuan, J., Brereton, G. J., & Murillo, M. S. (2021). Data-driven prediction of the equivalent sand-grain height in rough-wall turbulent flows. *Journal of Fluid Mechanics*, *912*, A8. https://doi.org/10.1017/jfm.2020.1085

Kim, J., Moin, P., & Moser, R. (1987). Turbulence statistics in fully developed channel flow at low reynolds number. *Journal of Fluid Mechanics*, *177*, 133–166. https://doi.org/10.1017/S0022112087000892

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, *807*, 155–166. https://doi.org/10.1017/jfm.2016.615

Lupo Pasini, M., Yin, J., Li, Y. W., & Eisenbach, M. (2021). A scalable algorithm for the optimization of neural network architectures. *Parallel Computing*, *104-105*, 102788. https://doi.org/https://doi.org/10.1016/j.parco.2021.102788

Moody, L. F. (1944). Friction factors for pipe flow. *Trans. Asme*, *66*, 671–684.

Napoli, E., Armenio, V., & De Marchis, M. (2008). The effect of the slope of irregularly distributed roughness elements on turbulent wall-bounded flows. *Journal of Fluid Mechanics*, *613*, 385–394. https://doi.org/10.1017/S0022112008003571

Nikuradse, J. (1933). Laws of flow in rough pipes. *VDI Forschungsheft*, 361.

Nilpueng, K., & Wongwises, S. (2015). Experimental study of single-phase heat transfer and pressure drop inside a plate heat exchanger with a rough surface. *Experimental Thermal and Fluid Science*, *68*, 268–275. https://doi.org/https://doi.org/10.1016/j.expthermflusci.2015.04.009

Orlandi, P., & Leonardi, S. (2006). DNS of turbulent channel flows with two- and three-dimensional roughness. *Journal of Turbulence, 7*, N73. https://doi.org/10.1080/14685240600827526

Parish, E. J., & Duraisamy, K. (2016). A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, *305*, 758–774. https://doi.org/https://doi.org/10.1016/j.jcp.2015.11.012

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. *NIPS-W*.

Peeters, J., & Sandham, N. (2019). Turbulent heat transfer in channels with irregular roughness. *International Journal of Heat and Mass Transfer*, *138*, 454–467. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2019.04.013

Sandberg, R. D., & Zhao, Y. (2022). Machine-learning for turbulence and heat-flux model development: A review of challenges associated with distinct physical phenomena and progress to date. *International Journal of Heat and Fluid Flow*, *95*, 108983. https://doi.org/https://doi.org/10.1016/j.ijheatfluidflow.2022.108983

Sanhueza, R. D., Akkerman, I., & Peeters, J. W. (2023). Machine learning for the prediction of the local skin friction factors and nusselt numbers in turbulent flows past rough surfaces. *International Journal of Heat and Fluid Flow, 103*, 109204. https://doi.org/https://doi.org/10.1016/j.ijheatfluidflow.2023.109204

Sanhueza, R. D., Smit, S. H., Peeters, J. W., & Pecnik, R. (2023). Machine learning for RANS turbulence modeling of variable property flows. *Computers & Fluids*, *255*, 105835. https://www.sciencedirect.com/science/article/pii/S0045793023000609

Schultz, M. P., & Flack, K. A. (2009). Turbulent boundary layers on a systematically varied rough wall. *Physics of Fluids*, *21*(1), 015104. https://doi.org/10.1063/1.3059630

Thakkar, M., Busse, A., & Sandham, N. (2018). Direct numerical simulation of turbulent channel flow over a surrogate for nikuradse-type roughness. *Journal of Fluid Mechanics, 837*. https://doi.org/10.1017/jfm.2017.873

Thakkar, M., Busse, A., & Sandham, N. (2017). Surface correlations of hydrodynamic drag for transitionally rough engineering surfaces. *Journal of Turbulence*, *18*(2), 138–169. https://doi.org/10.1080/14685248.2016.1258119

Ventola, L., Robotti, F., Dialameh, M., Calignano, F., Manfredi, D., Chiavazzo, E., & Asinari, P. (2014). Rough surfaces with enhanced heat transfer for electronics cooling by direct metal laser sintering. *International Journal of Heat and Mass Transfer, 75*, 58–74. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2014.03.037

Weatheritt, J., & Sandberg, R. (2016). A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship. *Journal of Computational Physics*, *325*, 22–37. https://doi.org/https://doi.org/10.1016/j.jcp.2016.08.015

Wong, T.-T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition, 48*(9), 2839–2846.

Yuan, J., & Piomelli, U. (2014). Estimation and prediction of the roughness function on realistic surfaces. *Journal of Turbulence, 15*(6), 350–365. https://doi.org/10.1080/14685248.2014.907904

5

OPTIMIZATION OF DIMPLED SURFACES FOR HEAT TRANSFER ENHANCEMENT

Dimpled surface designs are known to be effective at enhancing convective heat transfer. However, optimizing these surfaces can be challenging due to the large parameter space created by the different combinations between geometrical features. In this chapter, we combine a machine learning framework with a GPU-accelerated DNS solver to quickly assess the performance of a very large number of surface configurations, and to identify optimal designs. Our neural network can be trained to predict 2-D images with the local Nusselt numbers of rough surfaces within a few hours (in a single GPU), based on their original height maps. During evaluation, our neural network coupled with our parameterized geometrical formulation can evaluate one million dimpled surface designs in less than 45 minutes using a 64-core CPU architecture; with a low RAM memory footprint per core. Moreover, the GPU-accelerated DNS solver can calculate the Nusselt number of a rough surface within a few hours as well. The study considers a diverse parameter space including dimples with multiple depth profiles, major radiuses, corner effects, and inclination angles. To predict optimal designs, a basic reinforcement loop is created. In the first stage, only randomly chosen dimpled surface designs are selected as training data. The Nusselt numbers for each design are extracted from Direct Numerical Simulations (DNS), performed by the GPU-accelerated turbulent flow solver. Then, a convolutional neural network is trained, and different surface designs in our parameter space are evaluated. In order to advance the reinforcement learning loop, additional DNS cases are run for the optimal predicted surface, and other closely related geometrical variations. After adding these new DNS cases to the training set, the neural network is re-trained, and the process is repeated. Starting from the first iteration of the reinforcement learning loop, our results

This chapter has been published in the International Journal of Heat and Mass Transfer, volume 251, p. 127313, 2025.

shows that machine learning can predict remarkably optimized dimpled surface designs, with high Nusselt numbers verified through DNS. Moreover, we find that machine learning chooses dimple configurations that enhance the interaction between roughness elements, even if other dimples with shorter radius (and equal depth) have more heat transfer area. The optimal surface has elongated dimples with opposite inclination angles, which create a zig-zag pattern for the flow near the walls. Additionally, we have shown that at different Reynolds numbers, the optimal geometry is different as well. We analyze other plausible optimal dimpled surface designs within our parameter space, and we find that machine learning correctly identified the adequate parameters to maximize heat transfer. Therefore, we conclude that machine learning is a highly effective tool to identify optimized designs for convective heat transfer enhancement.

5.1. INTRODUCTION

Turbulent flows past rough surfaces can be found in different engineering applications. In most cases, rough surfaces tend to increase the drag resistance of transportation systems, leading to higher energy losses and fuel consumption. However, certain categories of rough surfaces can produce favorable results, such as enhancing heat transfer while only producing a modest increase in pressure losses (Chung et al., 2021; Dipprey & Sabersky, 1963). Therefore, it is possible to design special patterns for rough surfaces that maximize heat transfer inside engineering equipment. Maximizing the mean Nusselt number $\overline{Nu}$ of a rough surface is important for engineering applications, since heat transfer equipment can be made smaller overall. Different families of rough surfaces have been studied to accomplish this goal, such as riblets (Walsh & Weinstein, 1979) or dimples (P. M. Ligrani et al., 2003). Under special conditions, it has been shown that it is even possible to increase the net ratio between the Stanton number ($\overline{St}$) and the skin friction factor ($\overline{C_f}$) of a rough surface (Chung et al., 2021; P. M. Ligrani et al., 2003). An extensive review of rough surface patterns for heat transfer enhancement can be found in (P. M. Ligrani et al., 2003; P. Ligrani, 2013; Rouhi et al., 2022). In many studies, it has been noted that dimples have a high potential to increase heat transfer in a system (Afanasyev et al., 1993; Choi et al., 2023; Kumar et al., 2017; P. Ligrani, 2013; Rao et al., 2020). Moreover, dimpled surfaces are easy to manufacture, and different modifications can be considered, such as elliptical elements, rotated shapes (Rao, Feng, et al., 2015), vortex generators (Jeong et al., 2019; Xia et al., 2014). Other studies have further considered elliptical protrusions (Xie et al., 2018), trapezoidal geometries (Dagdevir et al., 2019), staggered arrangements (Abdus Samad & Kim, 2010), teardrop shapes (Rao, Li, & Feng, 2015), tubes with large dimples or protrusions (Cheraghi et al., 2020; Li et al., 2016).

One challenge while optimizing dimpled surface designs is that the parameter space created by the combinations between all geometrical features is very large. It is well-known that modifications in the dimple shapes can lead to substantially different results (P. M. Ligrani et al., 2003; P. Ligrani, 2013). Therefore, employing machine learning is necessary to assess the heat transfer performance of different dimple shapes. The current study uses a previously developed neural network architecture (Sanhueza et al., 2023), which is trained using existing DNS data, and it is capable of scanning the height map of a rough surface, and then predicting the local distribution of Nusselt numbers $Nu(x,z)$ or skin friction factors $C_f(x,z)$. While traditional convolutional neural networks must

process an entire image for each predicted scalar quantity, our neural network architecture (Sanhueza et al., 2023) can recycle the results of intermediate convolutional layers, and thus it is capable of generating 2-D maps with all $Nu(x,z)$ predictions after processing an image only once. Theoretically, this reduces the time and space complexity of the problem from quadratic to linear complexity, during both the training and evaluation stages. In our practical application, we indeed observe that the wall clock times for the training and evaluation of the neural network are reduced by several orders of magnitude. An example showcasing our convolutional neural network architecture, together with the filtering methodology previously developed (Sanhueza et al., 2023), can be found in this Github repository (Diez, 2025).

Thanks to the efficiency of the machine learning system, it becomes possible to create a reinforcement-learning study, where millions of rough surface designs are evaluated in a short time searching for optimal designs. Within this context, one important advantage of machine learning over gradient-based algorithms like the discrete adjoint method, is that the impact of large discrete changes in surface features can be assessed directly, without relying on infinitesimal gradients. Estimating the gradients with respect to changes in surface features is computationally expensive (Garai & Murman, 2021), especially for DNS with large grid sizes. Moreover, gradient-based optimizers also risk running into other numerical issues, such as convergence to local minima close to the starting configuration. About the usage of other optimization methods, we highlight that in Section 5.3, our current machine learning framework is able to find optimized dimpled surfaces using only 20 training samples, which is less than the number of degrees of freedom in the system. Classical optimization methods would require hundreds, or even thousands, of surface design evaluations (through DNS) before converging.

To further strengthen our computational framework, we use a GPU-accelerated DNS solver written in Fortran, allowing for fast and accurate generation of high-fidelity data. The GPU-based DNS solver is used both for generating training data, and for verifying the heat transfer performance of selected dimpled surface designs. In general, GPU-based DNS solvers can be substantially faster than even multicore CPU algorithms (Salvadore et al., 2013). Therefore, this work combines for the first time the neural network architecture developed by (Sanhueza et al., 2023) with a GPU-accelerated DNS solver to generate optimized dimple surface designs. This chapter is organized as follows: in Section 5.2, the methodology of the study will be presented, along with the parameter-space created for the rough surfaces. Then, in Section 5.3, the results of the study are described, which is followed by the conclusions in Section 5.4.

5.2. METHODOLOGY

5.2.1. GEOMETRICAL VARIATIONS

During the current study, a diverse collection of dimpled surface designs is considered, including variations in different types of relevant features. A schematic representation of the geometrical variations available for every dimple can be found in Figure 5.1. In this scheme, different shape modifications are highlighted, such as the radial profiles (R_{1-4}), dimple curvature (C_{a-b}), inclination angle (α), and depth profile (D). The values considered for each of these parameters are listed in Table 5.1. Regarding the depth profiles

shown in Figure 5.1(c), each spline is internally controlled by an anchor point located at $R(\theta)/2$. The first profile (blue) is a nearly sinusoidal shape, which resembles a classical dimple study, whereas the second profile (red) has a sharper inclination angle. Either of these shapes can be chosen during the machine learning optimization study. As a side-note, more shape variations were initially considered. However, during a preliminary study, intermediate depth profiles only produced gradual variations in our subsequent DNS results, and the optimum Nusselt number $\overline{Nu}$ was found for one of the two extreme depth profiles (round/sharp). Regarding the dimple curvature effects shown in Figure 5.1(b), the "elliptical" shape is obtained by considering four different ellipses, with major radiuses R_{1-4} spaced at 90° intervals. In contrast, the "circular corner" effect uses the smaller dimple radius to draw a circular arc at 90°. From a physical perspective, this circular corner effect can increase the windward area of the dimples, where the maximum heat transfer typically occurs. However, this profile could also have the opposite effect, and expand the region with recirculation inside the dimples, which is detrimental for heat transfer. Therefore, the variations in the curvature profiles pose an interesting challenge for the machine learning study. The inclination angle (α) is physically relevant as well, since inclined dimples can enhance vortex generation compared to spherical dimples (Isaev et al., 2023; Rashidi et al., 2019). However, the machine learning system must determine the optimal inclination angle (α), and how different dimple shapes will affect vortex interactions. Finally, regarding the depth of the dimples d_{ref}, this parameter is kept at a fixed value of $d_{ref} = \delta/10$, where δ is the half channel height: $\delta = L_y/2$. If d_{ref} is changed, our DNS analysis showed that the Nusselt number $\overline{Nu}$ tends to increase monotonically, even for very large d_{ref}/δ ratios. Such DNS cases require more grid cells, and the identification of adequate initial conditions to avoid divergence. Good initial conditions are often obtained by running preliminar DNS cases with less pronounced geometrical features. However, the process of running additional simulations is computationally expensive, and it requires the formulation of empirical selection criteria. Therefore, all DNS simulations were performed only for $d_{ref} = \delta/10$. As a general note, our dimpled surfaces with $d_{ref} = \delta/10$ can display Nusselt numbers up to 53% higher than smooth wall channel flows under equal pressure losses (at $Re_\tau = 180$), and the simulations can start from almost any initial condition.

Table 5.1: Parameters considered for the dimpled surfaces. The variable α is the inclination angle for the dimples, whereas R_{1-4} are the four major radiuses for each dimple. The parameters C_{a-b} are the two different curvature variations, and D is the depth profile for every dimple. The influence of the previous parameters in the dimple geometry is sketched in Figure 5.1.

Variable	Alternatives						
α	-45°	-30°	-15°	0°	15°	30°	45°
R_{1-4}	30%	50%	70%				
C_{a-b}	Elliptical	Rounded corner					
D	Rounded	Sharp					

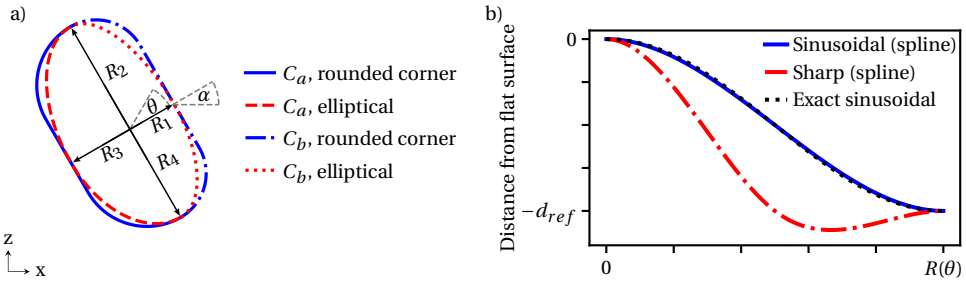


Figure 5.1: Schematic representation of the parameters considered for the dimples. The sketch (a) highlights the location of the radiuses R_{1-4} within the dimples and their inclination angle (α) with respect to the $x-z$ plane. The legend of subplot (a) indicates the different curvature effects available for the dimples. Finally, the subplot (b) shows the two possible depth profiles for the dimples. Within this plot, d_{ref} is the reference depth for the dimples, whereas $R(\theta)$ is the local dimple radius at any given angle θ. Please note that $R(\theta)$ is interpolated from subplot (a) taking into consideration the local curvature effects.

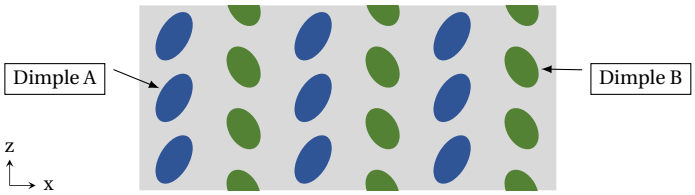


Figure 5.2: Representation of the staggered dimple arrangement for the simulations. The shape of each dimple (A/B) is given by the alternatives listed in Table 5.1.

Regarding the global dimple arrangements, the final study consists in optimizing two rows of staggered dimples that are periodically repeated, as is it shown in Figure 5.2. Each row of dimples has a unique shape, according to the alternatives listed in Table 5.1. The total number of configurations available for the system is approximately 3.5 million, after removing repeated entries due to periodicity and physical symmetry in the spanwise direction. The dimples within each row (in the z-direction) have identical shape, since a preliminary study showed that the flow across the dimples mainly interacts with other elements aligned in the streamwise direction. Moreover, adding more dimple variations in the spanwise direction (per row) exponentially increases the number of combinations available, and it was not found the change the outcome of preliminary ML optimization studies. Regarding the spacing between the dimples, all shapes are packed as closely as possible, while preserving the staggered grid arrangement. The center of every dimple was chosen to be the average position of the region with a position deeper than 75% of the nominal depth (d_{ref}). This formula does not change the center of uniform dimples, and it ensures that the deepest parts of asymmetrical dimples follow a staggered arrangement. This is important to reduce flow obstruction, since the deepest area of the dimples has a large impact in flow circulation, and the regions with the highest amount of heat transfer are usually located in front of them. Therefore, the current formula ensures that the dimples have a good alignment within our dataset, although other methods could be tested in the future.

5.2.2. MACHINE LEARNING FRAMEWORK

The overall framework of the machine learning study corresponds to a basic reinforcement-learning loop, as it is shown in Figure 5.3. In this framework, a DNS database is generated first (5.3.A), containing purely random dimpled surfaces. Then, a neural network is trained (5.3.B) to predict the local Nusselt numbers $Nu(x,z)$ distributed across the rough surfaces. Using the newly trained neural network, the mean Nusselt numbers $\overline{Nu}$ are predicted for all possible dimpled surface combinations (5.3.C) according to Table 5.1. Considering the best performing dimpled surface (5.3.D), new, yet closely related designs are generated randomly, according to the sub-steps of (5.3.E). Additional channel flow cases are run for each of these new dimpled surfaces, and their results are added to the DNS database (5.3.F), in order to repeat the reinforcement-learning loop. Beyond the framework presented in Figure 5.3, it is important to note that the size of the neural network trained in step (5.3.B) was increased starting from the third iteration of the reinforcement learning loop. This modification was necessary to fit the increasing amount of training data available, since the size of our DNS database approximately triples by the third stage of the reinforcement learning study. Otherwise, the neural network would display a relatively high bias even when performing predictions for the training dataset. The details of the neural network architecture are discussed at the end of this sub-section.

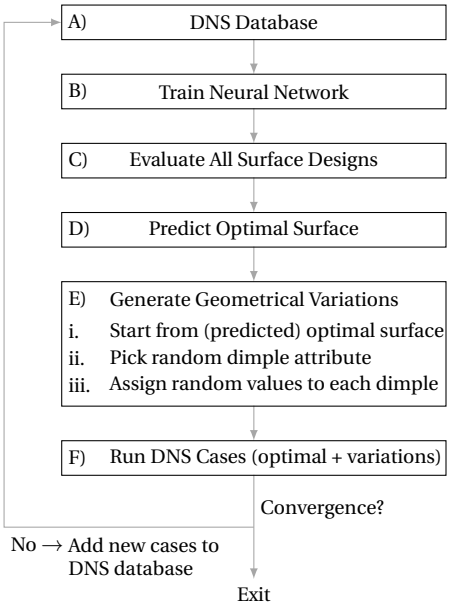


Figure 5.3: Steps of the basic reinforcement-learning loop for the current study.

In steps (i-iii) of Figure 5.3.E, the methodology to choose variations of the optimal dimpled surface configuration predicted by machine learning is described. In this methodology, the dimple configurations are changed, by picking one parameter, and giving it random values for both dimples. The main benefit of this approach is that the influence of each physical feature is tested separately. If more parameters were changed at the same

time, many configurations would have a distorted shape, which is physically far from the optimal predicted configuration. For example, the dimples configurations can be highly sensitive to changes in their depth profile, or their radii. Therefore, the current methodology establishes a better trade-off between exploring new design characteristics, and keeping a shape that is reasonably related to the best predicted configuration. The total number of possible variations is 93, according to Table 5.1. Only 20 of these random dimple configurations are sampled in every stage of the reinforcement learning loop. Therefore, the machine learning system must still use sparse information to make predictions at any given stage.

Regarding the details of the neural network architecture, the system is based on the optimized convolutional neural network (CNN) architecture described in (Sanhueza et al., 2023). This machine learning system is able to scan the height map of a rough surface $H(x,z)$ and predict its local Nusselt number distribution $Nu(x,z)$ with linear time complexity, during both the training and evaluation stages. In our target application, this is thousands of times faster than a traditional convolutional neural network, which must process an entire image for every scalar quantity predicted. Predicting the local $Nu(x,z)$ values can be beneficial, because it forces machine learning to build a predictive model that accounts for local flow effects. Moreover, this methodology also provides a one-to-one ratio between the input data ($H(x,z)$) and the predicted quantities ($Nu(x,z)$) for the neural network, which otherwise would be very unbalanced if $\overline{Nu}$ was predicted directly. Regarding traditional surface metrics to predict $\overline{Nu}$, such as the skewness or the effective slope, these quantities only correspond to averaged values, and they do not take into account the alignment between the roughness elements (Forooghi et al., 2018). During the current study, it was observed that identical roughness elements with optimized alignment could substantially increase heat transfer. Therefore, traditional surface metrics do not form a good basis for the optimization study. Our machine learning framework to predict local $Nu(x,z)$ distributions is well suited to take into account how local flow effects influence heat transfer.

Table 5.2: Neural network architectures to predict the local Nusselt numbers $Nu(x,z)$. The parameters C_{in}, C_{out}, K and D refer to the number of input channels, output channels, kernel size, dilation and the presence of an activation function respectively. The total number of trainable parameters is 4,461 for the initial neural network, and 8,161 for the larger ML model.

Layer	C_{in}	C_{out}	K	D	AF
Initial neural network					
DSC	1	20	3	1×1	Yes
DSC	20	20	3	2×2	Yes
DSC	20	20	3	4×4	Yes
DSC	20	20	3	8×8	Yes
DSC	20	20	3	16×16	Yes
DSC	20	20	3	32×32	Yes
Conv. 1D	20	1	1	-	-
Large neural network					
DSC	1	20	3	1 × 1	Yes
DSC	20	20	3	1 × 1	Yes
DSC	20	20	3	2 × 2	Yes
DSC	20	20	3	2 × 2	Yes
DSC	20	20	3	4 × 4	Yes
DSC	20	20	3	4 × 4	Yes
DSC	20	20	3	8 × 8	Yes
DSC	20	20	3	8 × 8	Yes
DSC	20	20	3	16 × 16	Yes
DSC	20	20	3	16 × 16	Yes
DSC	20	20	3	32 × 32	Yes
Conv. 1D	20	1	1	-	-

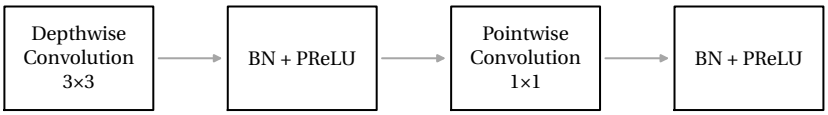


Figure 5.4: Schematic representation of the depthwise separable convolution (DSC) modules mentioned in Table 5.2. The abbreviation BN refers to the 2-D batch normalization operations applied before each PReLU activation function.

The overall neural network architecture for the current study is described in Table 5.2. Here, two neural networks are presented. The first network has a reduced number of layers, which is used during the first two optimization stages of the study, due to the

smaller amount of training data available in the DNS database. In the third reinforcement learning iteration, a larger neural network is employed, in order to fit the increasing amount of training data. The details of each DSC module mentioned in Table 5.2 are sketched in Figure 5.4. This architecture follows the principles described in (Sanhueza et al., 2023), although the number of activation functions inside the network was reduced. This minor modification was made to reduce the number of trainable parameters, and because small empirical differences were found in the context of the current study. The neural networks listed in Table 5.2 have a total of 4,461 and 8,161 trainable parameters, for the smaller and the larger neural network respectively.

5.2.3. GPU-ACCELERATED DIRECT NUMERICAL SIMULATIONS (DNS)

In this study, the dimpled surfaces are simulated as rough walls within rectangular channel flows. The numerical framework is primarily based on the dimensionless form of the incompressible Navier-Stokes equations:

$$\nabla \cdot \mathbf{u} = 0, \tag{5.1}$$

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\nabla p + \frac{1}{Re_\tau}\nabla^2 \mathbf{u} + S_f, \tag{5.2}$$

$$\partial_t T + (\mathbf{u} \cdot \nabla)\,T = \frac{1}{Re_\tau Pr}\nabla^2 T + S_q, \tag{5.3}$$

In eqs. (5.1, 5.2, 5.3), the variables $\mathbf{u}$, P, T refer to the fluid velocity vector, pressure and temperature respectively. The parameter Re_τ is the friction Reynolds number of the fluid, whereas Pr is the molecular Prandtl number. The constants S_f and S_q are source terms that induce fluid motion and heat transfer in their respective equations (Busse et al., 2015; J. Kim et al., 1987; Orlandi & Leonardi, 2006; Peeters & Sandham, 2019; Thakkar et al., 2017). The geometry considered for the study is a planar channel flow with dimpled walls, as it is shown in Fig. 5.5. In this scheme, $H(x,z)$ is the height function for the dimpled surface, the y-coordinate is the wall-normal direction, and L_y corresponds to the full-channel height. Periodic boundary conditions are considered in the streamwise (x) and spanwise directions (z). The challenge of the heat transfer optimization problem is to maximize the mean Nusselt number ($\overline{Nu}$), which is given by:

$$\overline{Nu} = \frac{\overline{q_w}}{\lambda\,(T_b - T_w)/L_y}, \tag{5.4}$$

where $\lambda = 1/(Re_\tau\ Pr)$ is the dimensionless conductivity of the fluid, T_w is the wall temperature, and T_b is the bulk temperature of the fluid. As it can be observed, maximizing the Nusselt number of the system ($\overline{Nu}$) is equivalent to decreasing the temperature difference ($T_b - T_w$), since all other parameters are fixed in the current study. The mean heat flux $\overline{q_w}$ is also fixed, because the energy balance must be in equilibrium with respect to the volumetric source term S_q.

From the global simulation settings, it is important to highlight that the channel flows operate with a constant mean pressure loss $\overline{\Delta P}$, since the friction drag losses at the rough walls are compensated by the momentum source term S_f. As a consequence, in this study, the Nusselt number for the channel flow is maximized without changing the average

pressure losses. This marks a contrast with respect to other formulations, where the highest Nusselt number might be found for configurations with very high pressure losses. Another implication of the current settings is that the skin friction factor $\overline{C_f}$ is not a direct target of optimization for the study.

In practical terms, our optimization study resembles cases like the design of a heat exchanger. We seek to provide a constant heating load Q with fixed pressure losses ΔP, and the objective is to maximize the Nusselt number $\overline{Nu}$ to ensure that the system can operate with the lowest temperature difference: $(T_b - T_w)$. Here, the drag losses should be measured in terms of the hydraulic pumping power, which is proportional to $\Delta P\ U_b$, among other fixed parameters (L_x, L_z, etc.). Based on these settings, dimpled surfaces with higher hydraulic resistance will have a lower bulk velocity (U_b) and Reynolds number (Re_b), and hence their pumping power will not be greater. Despite this fact, since the Nusselt number $\overline{Nu}$ is highly correlated with Re_b, surfaces with high hydraulic resistance will be subject to a trade-off, where their $\overline{Nu}$ values are penalized by their lower bulk velocity. Therefore, the results of the optimization study are physically-relevant.

In the literature, most studies tend to address a different challenge, which is maximizing the thermal efficiency $(Nu/Nu_0)/(C_f/C_{f,0})$ with respect to a smooth wall (sub-index 0); for a fixed bulk Reynolds number (Re_b) (P. M. Ligrani et al., 2003; P. Ligrani, 2013). While surfaces with high thermal efficiency are valuable, this metric is not directly applicable to this study, because the changes in the skin friction factor (C_f) will modify the pressure losses ΔP, and hence the channel flow will reach another equilibrium with different Re_b. As a result, we highlight that maximizing $\overline{Nu}$ is the right optimization target for the current study (with fixed ΔP and Q values).

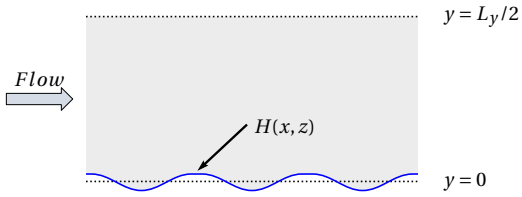


Figure 5.5: Scheme of the channel flow geometry, with the height function $H(x,z)$ representing the dimpled surfaces. The wall-normal direction is given by the y-coordinate, where x is the streamwise direction. The z-coordinate, perpendicular to the image, is the spanwise axis. The plane $y = 0$ is coincident with the average height for the dimpled surface ($H(x,z)$). The variable L_y is the full channel height, and thus $y = L_y/2$ is the symmetry plane at the channel center.

Regarding the discretization and simulation parameters, the study is performed considering air with $Pr = 0.71$, $S_f = 1$, $S_q = 1$ and $L_y = 2$. The friction Reynolds number (Re_τ) can be either 180 or 395, depending on the simulation settings. The grid size is $\left(N_x \times N_y \times N_z\right) = (800 \times 256 \times 400)$ in each spatial direction for the simulations at $Re_\tau = 180$, whereas the $Re_\tau = 395$ use a grid size of $\left(N_x \times N_y \times N_z\right) = (1920 \times 592 \times 960)$. The size of the channel flows in the streamwise (L_x) and spanwise directions (L_z) varies between $L_x = [5.66,\ 7.50]$ and $L_z = [2.82,\ 4.13]$, depending on the size of the dimples simulated. These domain sizes are consistent with other works about DNS for turbulent flows past rough surfaces (Leonardi et al., 2015; Peeters & Sandham, 2019; Thakkar et al.,

2018). In order to simulate the behavior of the rough surfaces, the immersed boundary method (IBM) is used. Our IBM implementation is inspired by the Fadlun scheme (Fadlun et al., 2000), although we use a ghost point inside the solid instead of forcing the first (fluid) point above the rough surfaces. Extensive validation benchmarks were performed, obtaining identical results as (Peeters & Sandham, 2019). Due to the refined grid, all the current DNS cases have dimensionless cell sizes in the range $\Delta x^+ < 1.69$, $\Delta z^+ < 1.86$, and a uniform mesh size of $\Delta y^+ \approx 0.9$ is kept near the rough surfaces. The mesh size is small in the x and z directions, since the dimples have a significant curvature, and this helps to reduce the distance at which interpolation points are considered. The grid size further complies with the basic discretization requirements of similar simulations (Busse et al., 2015; Peeters & Sandham, 2019; Thakkar et al., 2017), which can be expressed as $\left(\Delta x^+,\ \Delta y^+,\ \Delta z^+\right) < (5,\ 1,\ 5)$. Other discretization constraints related to the length-scale of the rough surfaces do not apply to the current study, since the dimples are substantially larger than the smallest grid scales. More details about the validation of the GPU-based DNS solver can be found in C.1.

From a fluid mechanics perspective, it is important to mention that all DNS cases with dimpled surfaces operate in turbulent regimes. Both smooth wall and rough surface cases display turbulent flow behavior at $Re_\tau = 180$ for moderate roughness heights (Busse et al., 2015; Peeters & Sandham, 2019; Thakkar et al., 2017).

About the software implementation, our GPU-accelerated DNS solver is written in Fortran, using GPU-aware MPI and OpenACC for cross-platform compatibility. The numerical algorithm uses the fractional-step method, following the methodology used by (Peeters & Sandham, 2019). Finite difference expressions are used for the spatial discretization of the Navier-Stokes equations, whereas the temporal discretization is given by an Adams-Bashford scheme. The immersed boundary method uses ghost points along fixed Cartesian directions, as previously mentioned. To fulfill the continuity equation, a pressure-Poisson equation is implicitly solved using spectral FFT-based methods, achieving $\mathcal{O}(n\ log\ n)$ time complexity for large-scale applications. In the Poisson solver, Fourier transformations are applied in the streamwise (x) and spanwise (z) directions, whereas the wall-normal direction (y) retains its finite difference discretization and a tridiagonal solver is used instead. To ensure high-performance in multi-GPU configurations, the computational domain is divided into 1D slabs along the y-direction for each MPI task. This implies that all FFT operations along the x-z directions are local to each GPU device. To solve tridiagonal equations along the y-direction, we incorporate a parallel solver, based on the parallel cyclic reduction (PCR) algorithm (Sanhueza et al., 2025; Yang et al., 2023).

5.2.4. SCALABILITY ANALYSIS

Regarding the scalability of our optimization framework, only the running times of the DNS solver are affected by the higher Reynolds number (Re_τ). The input images passed to the machine learning system, with the local Nusselt number distributions $Nu(x,z)$, can be rescaled to a lower resolution similar to the DNS grid for $Re_\tau = 180$. In our experience, this resolution is enough to capture both the geometrical features of the dimpled surfaces and the changes in the local $Nu(x,z)$ distributions, even for flows at higher Reynolds numbers. Therefore, the running times of our machine learning framework are constant

for flows at different Reynolds numbers. In the case of the DNS solver, the Poisson solver tends to be the main performance bottleneck for large-scale simulations, since the remaining subroutines of the DNS solver either perform halo exchanges or local kernel computations. Despite these challenges, our DNS code incorporating a parallel tridiagonal solver is able to achieve strong scalability (by grid size) when comparing simulations at either $Re_\tau = 180$ or $Re_\tau = 395$. Detailed benchmarks about the scalability of DNS solvers with PCR algorithms can be found in (Costa, 2025; K.-H. Kim et al., 2023; Sanhueza et al., 2025; Yang et al., 2023).

5.3. RESULTS

5.3.1. OPTIMIZATION FOR DIMPLES SURFACES AT $Re_\tau = 180$

In order to initialize the reinforcement-learning loop, described in Figure 5.3, a DNS database with completely random dimple variations is generated first. The dimpled surfaces found in this initial database are plotted in Figure 5.6. Here, it can be verified that all the 20 DNS cases have unique patterns and different surface topologies. Using the GPU-accelerated DNS solver, the local Nusselt number distributions $Nu(x,z)$ for each of these 20 random surfaces are calculated. Based on these results, the smallest neural network described in Table 5.2 is trained. From the 20 DNS cases available, 18 DNS cases are used as training data, and 2 DNS cases are left as a small cross-validation set to monitor over-fitting. After training the neural network, approximately 3.5 million different surface designs are evaluated, seeking to maximize the mean Nusselt number $\overline{Nu}$. The best three performing configurations predicted by the neural network in the first reinforcement-learning generation are shown in Figure 5.7. Here, it can be seen that machine learning predicts that the optimal surface consists of two elongated dimples with opposite inclination angles (α). This result is very interesting, since none of the shapes in the training data (Fig. 5.6) had a similar pattern. The Nusselt numbers predicted by machine learning, and the DNS verification for the top-3 configurations are also shown in Figure 5.7. From these results, it can be noted that the ranking assigned to the dimpled surface designs is correct with respect to their Nusselt numbers $\overline{Nu}$ extracted from the DNS data, even though the neural network slightly under-predicts the mean Nusselt number $\overline{Nu}$ compared to the DNS results.

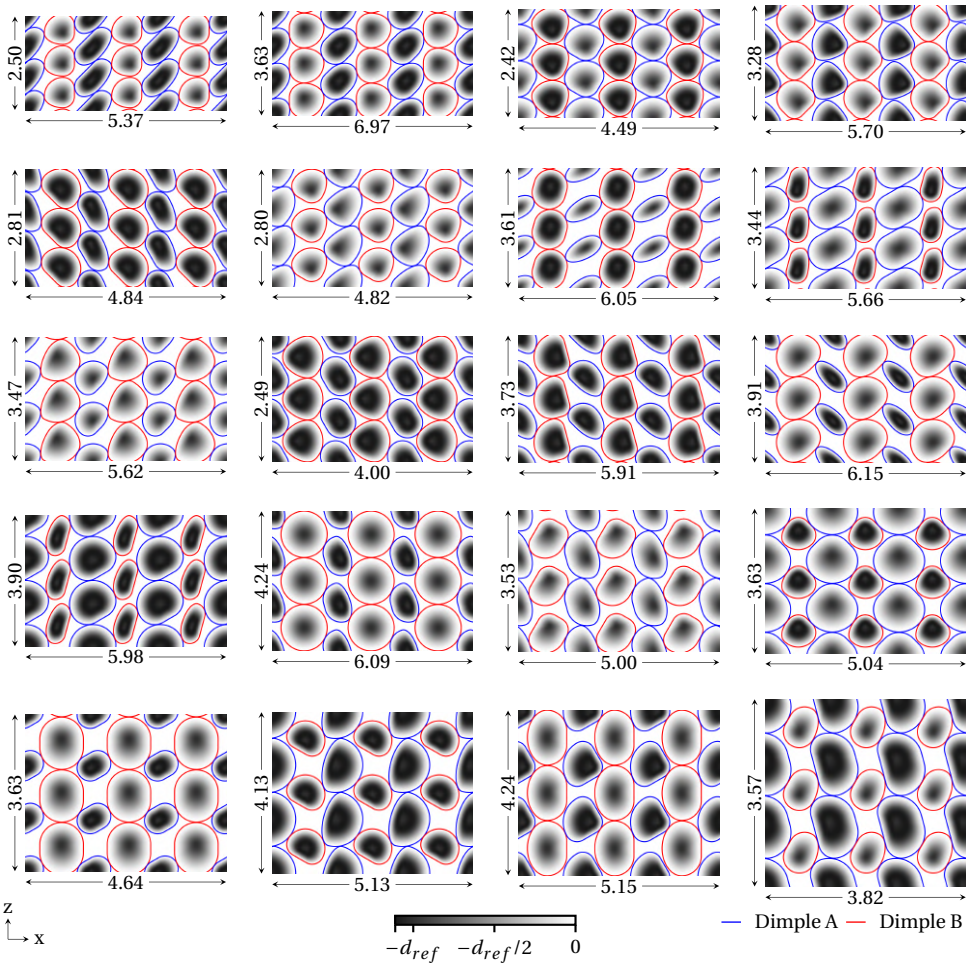


Figure 5.6: Dimpled surfaces found in the DNS database for the first generation of the reinforcement-learning framework. The sub-images are drawn at scale to represent their size in physical coordinates: ($L_x \times L_z$).

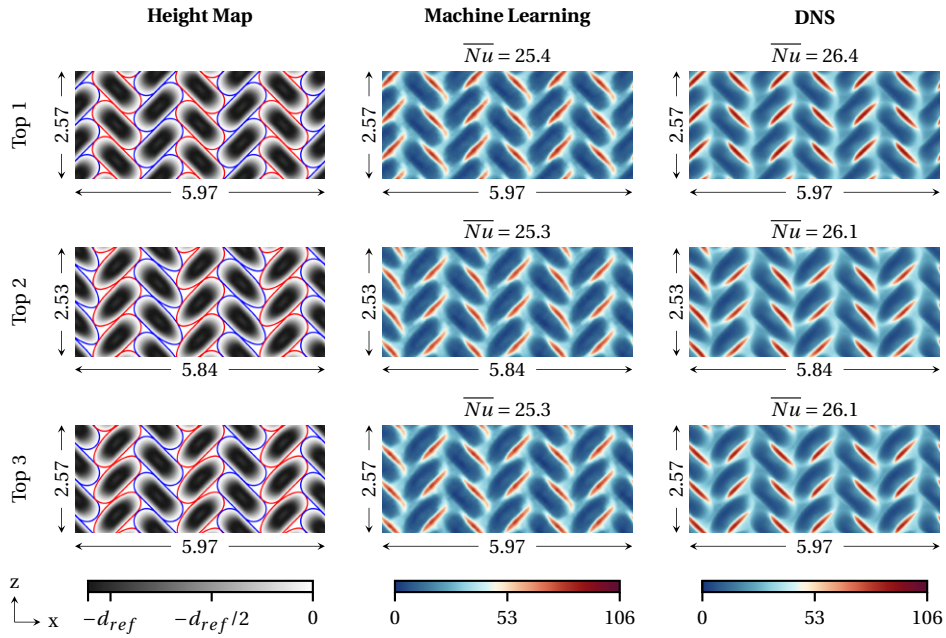


Figure 5.7: Top ranked dimpled surfaces found during the first generation of the reinforcement learning loop at $Re_\tau = 180$. The bulk flow moves from left to right.

For the second iteration of the reinforcement learning loop, new dimpled surfaces are generated, according to the methodology described in steps (i-iii) of Figure 5.3. The surfaces created can be found in Figure 5.8. Here, it can be noted that the new surfaces contain large variations from a physical perspective, despite only changing one parameter at a time. The neural network is thus re-trained using an extended dataset with 41 DNS cases: the 20 original surfaces shown in Figure 5.6, the top-1 ranked surface identified by machine learning (first row in Fig. 5.7), and the variations shown in Figure 5.8. After re-training the neural network, the results shown in Figure 5.9 are obtained. The main difference with respect to the first generation of the reinforcement learning loop is that inclination angle of the dimpled surfaces was changed from 45° to 30°. The variations in the top predicted configurations are minor, and they follow the same patterns observed back in Figure 5.7.

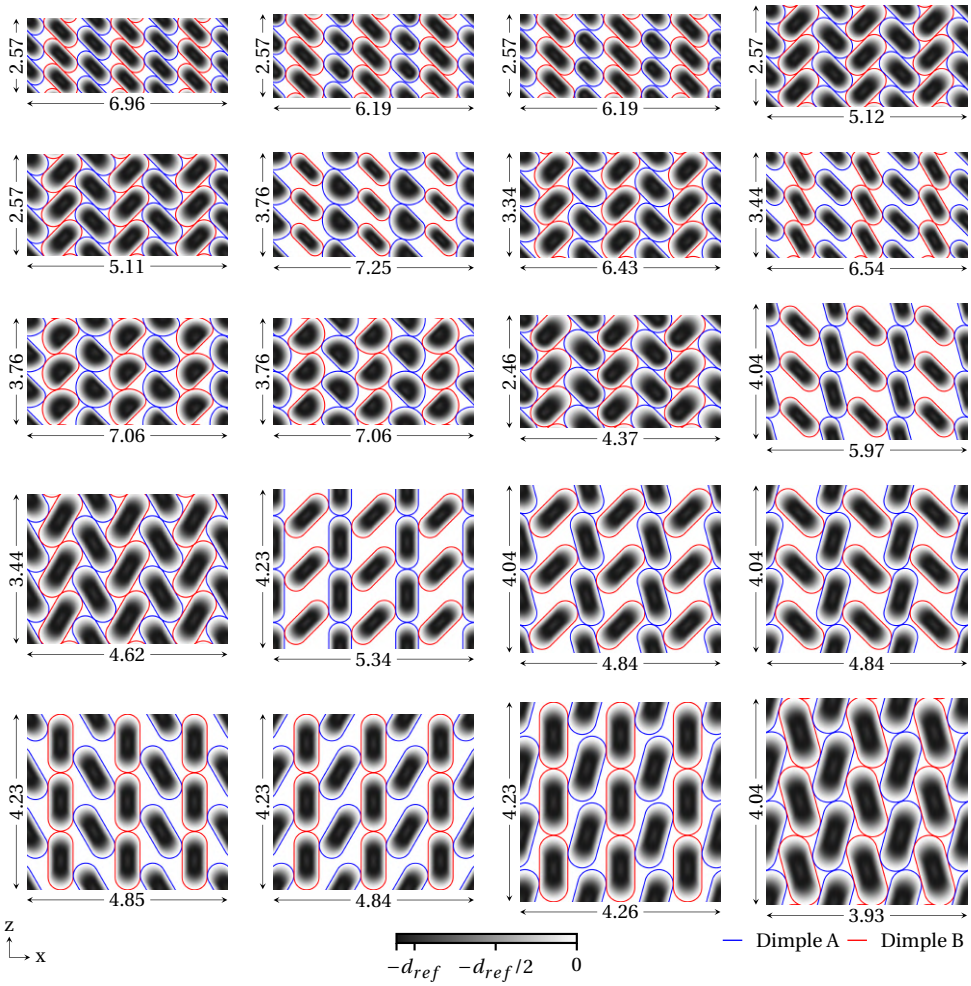


Figure 5.8: Additional dimpled surfaces added to the DNS database for the second generation of the reinforcement-learning framework.

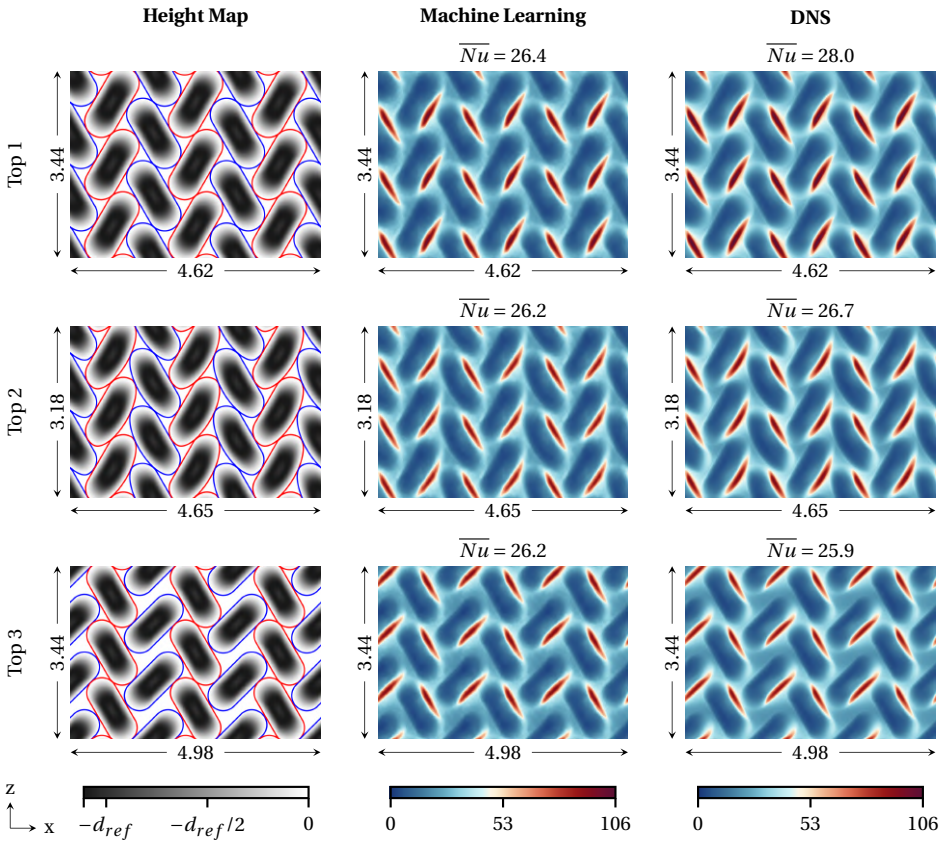


Figure 5.9: Top ranked dimpled surfaces found during the second generation of the reinforcement learning loop at $Re_\tau = 180$. The bulk flow moves from left to right.

In the third generation of the reinforcement-learning study, 20 additional surfaces are considered. Now the DNS database contains a total of 62 DNS cases, since the optimal design from the second generation (Fig. 5.9) is also added to the database. Due to the increased amount of training data, the size of the neural network was increased. Otherwise, the bias of the neural network can be higher, even for training samples. The results of the machine learning study are found in Figure 5.10. Here, it can be seen that the top predicted configurations have a very similar topology to the second reinforcement learning stage, although small variations in the curvature of the perimeter for the dimples are observed. All dimpled surfaces still have high potential for heat transfer, according to the DNS results. However, it is observed that the minor topological changes observed among the top dimpled surfaces do not further influence the outcome of the study, and the reinforcement-learning iterations are concluded.

To investigate which surface features contribute the most to heat transfer enhancement, Figure 5.11 presents a comparison between multiple variations from the optimal surface predicted in the second generation of the reinforcement learning study. Here, the geometry of the dimpled surfaces is shown along with their respective Nusselt numbers $\overline{Nu}$ extracted from DNS data. From this comparison, it can be noted that the configuration predicted by machine learning is substantially better than similar designs. This result indicates that the neural network correctly found an optimized dimpled surface design starting from low-quality random training data, which had little relation with the optimal patterns. Many reinforcement learning frameworks would require hundreds, if not thousands, of iterations before they can converge to an adequate design. In contrast, our machine learning system quickly found correct geometrical parameters, such as the inclination angle $\alpha = 30°$, corner profiles, depth and aspect ratio for the dimples among similar alternatives. Regarding these surfaces, it can also be highlighted that the dimples with a circular perimeter, or lower aspect ratio, are more closely packed and have more heat transfer area. Therefore, machine learning took into account the interactions between roughness elements while assessing the potential for heat transfer of different surface designs.

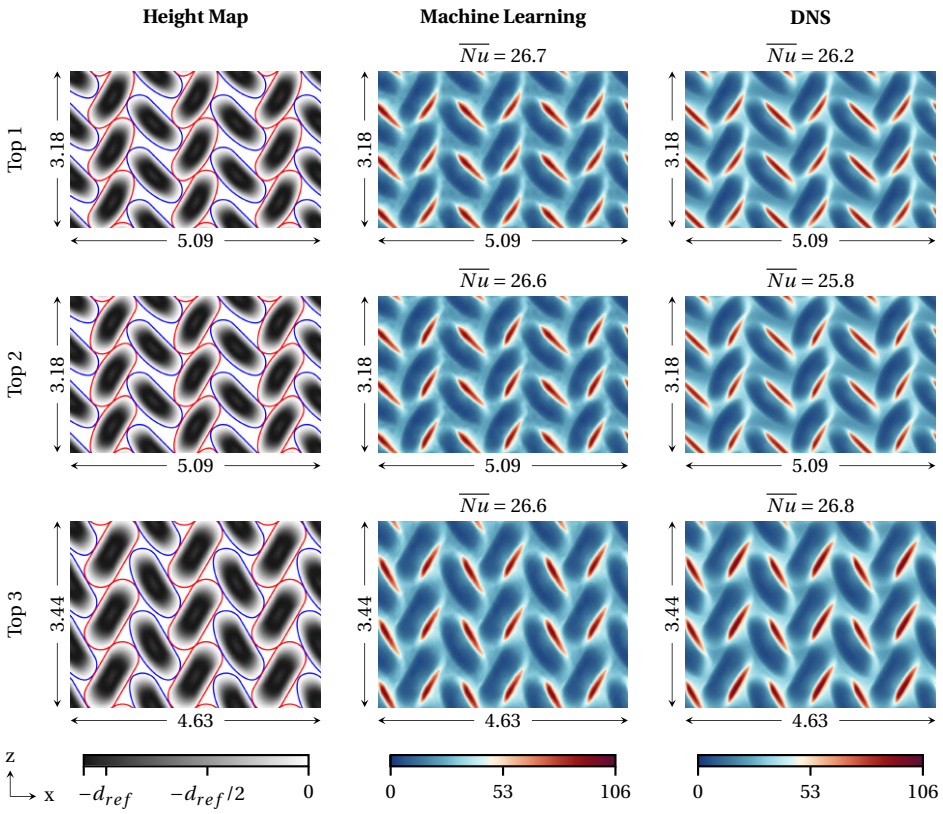


Figure 5.10: Top ranked dimpled surfaces found during the third generation of the reinforcement learning loop at $Re_\tau = 180$. The bulk flow moves from left to right.

Finally, it is interesting to investigate if the performance of the optimal dimpled surface can also be explained by investigating the local flow structures. Flow patterns are generally important for dimpled surfaces, since most heat transfer takes place in the windward faces, whereas backflow areas (with recirculation) have a negligible contribution. Due to this reason, the averaged Nusselt numbers $\overline{Nu}$ of dimpled surfaces are greatly dependent on any heat transfer enhancement effects at the windward faces, due to factors such as the shape or alignment of dimple elements. Based on this context, the higher Nusselt numbers for dimpled surfaces with opposite inclination angles can be attributed to the creation of a zig-zag pattern in the flow near the surfaces. This phenomenon is visualized in Figure 5.12, where the streamlines for the top surface from the second generation of the reinforcement learning loop (Figure 5.9) are compared with the dimples with circular perimeter shown in Figure 5.11 and other closely related geometrical variations. Here, it can be clearly observed that the dimples with opposite inclination angles have minimal fluid recirculation, since the streamlines inside each dimple follow a spiral pattern (Isaev et al., 2019), where the flow from previous dimples is deviated resulting in a large heat transfer rate in the frontal faces. In contrast, the circular dimples suffer from significant

flow recirculation in their interior (Chen et al., 2012; Elyyan et al., 2008; Turnow et al., 2012; Zhengyi Wang & Khoo, 2006), which is detrimental for local heat transfer. Another reason for the lower thermal performance of circular dimples is that the flow perturbations created by one element do not have a significant impact on the upcoming dimples. This phenomenon lowers the thermal performance of aligned roughness elements in general. Beyond the alignment of the roughness elements, the larger $\overline{Nu}$ values observed for dimples with sharper corners, or steeper depth profiles, is related to the increased size of the frontal area where the highest local Nusselt numbers $Nu(x,z)$ are found.

Regarding the influence of the dimple inclination angle, Figure 5.12 shows that dimples with a moderate inclination of 15° have a much higher degree of recirculation than the optimal configuration (30°). This is evidenced by the additional loops of the spiral vortices inside these dimples. Dimples with a higher inclination angle of 45° display minimal recirculation, yet it can be observed that the spiral vortices inside the dimples are highly distorted. Finally, the dimples in aligned arrangement at 30° present a drastically different flow pattern, where the flow tends to hover above the dimples, which is less than optimal for heat transfer. In summary, it can be observed that both the inclination angle of the dimples and the staggered arrangement have a large impact in the flow patterns for every configuration, and the resulting heat transfer.

Therefore, our machine learning framework can be a useful tool to predict surface designs that maximize heat transfer, and we provide a framework to iterate using additional results obtained from DNS simulations. Moreover, the local Nusselt numbers $Nu(x,z)$ predicted for each configuration serve as an additional verification tool, to check whether the distributions follow physically plausible patterns, or if the mean Nusselt number $\overline{Nu}$ are affected by spurious non-physical oscillations. Further design insights can also be obtained from $Nu(x,z)$ to predict optimal heat transfer patterns, or to understand the sensitivity of the neural network to changes in the dimple geometry.

To provide a broader context, the performance metrics reported by (P. M. Ligrani et al., 2003) and the Reynolds analogy factor ($2St/C_f$) (Bons, 2005) were computed for the top dimpled surface from the second generation of the machine learning study, and other canonical flow cases. These values are reported in Table 5.3, along with the dimensionless ratio between heat transfer (Q) and pumping power (W_p) for each configuration. Here, it can be observed that the top dimple configuration also has a significantly higher ratio $Q/W_p/(Q_0/W_{p,0})$. Furthermore, the metric $\overline{Nu}/\overline{Nu}_0/(C_f/C_{f,0})^{1/3}$ is also higher for the optimized surface. However, according to the Reynolds analogy factor and the thermo-hydraulic efficiency, the optimized surface performs less well. Thus, we conclude that the optimization method yields an improved ratio of thermal load to pumping power. The latter is the direct result of the Nusselt number being much higher for the top dimpled surface than it is for the circular dimple surface. Moreover, our channel flow simulations have fixed pressure losses ΔP, which implies that the bulk Reynolds number Re_b is subject to change. Therefore, the value reported by the thermo-hydraulic efficiency does not necessarily reflect the final performance of the system.

Finally, to emphasize that the computed flows at $Re_\tau = 180$ operate in the turbulent regime, Figure 5.13 presents a snapshot of the instantaneous velocity field U^+ at the plane $z = L_z/2$ for the top-performing dimpled surface from the second generation of the reinforcement learning study. Here, it can be seen that the velocity field displays the

Table 5.3: Thermal performance metrics for the highest performing dimpled surface found in the second generation of the reinforcement learning study at $Re_\tau = 180$, and other closely related alternatives. The sub-index "0" refers to a flat-plate channel case at identical bulk Reynolds number (Re_b) as the target configuration.

Case	$\frac{Q/W_p}{Q_0/W_{p,0}}$	$\overline{Nu}$	C_f	$\frac{2St}{C_f}$	$\frac{\overline{Nu}/\overline{Nu}_0}{(C_f/C_{f,0})^{1/3}}$	$\frac{\overline{Nu}/\overline{Nu}_0}{C_f/C_{f,0}}$
Top dimpled surface (gen. 2)	1.63	28.0	0.024	1.00	1.54	0.81
Circular dimples (same Re_b)	1.25	17.8	0.014	1.10	1.17	0.88
Smooth walls, $Re_\tau \approx 110$ (same Re_b)	1	13.2	0.0091	1.25	-	-
Smooth walls, $Re_\tau = 180$ (same ΔP)	-	18.1	0.0079	1.13	-	-

similar characteristics as turbulent flow snapshots reported in other studies (Peeters & Sandham, 2019).

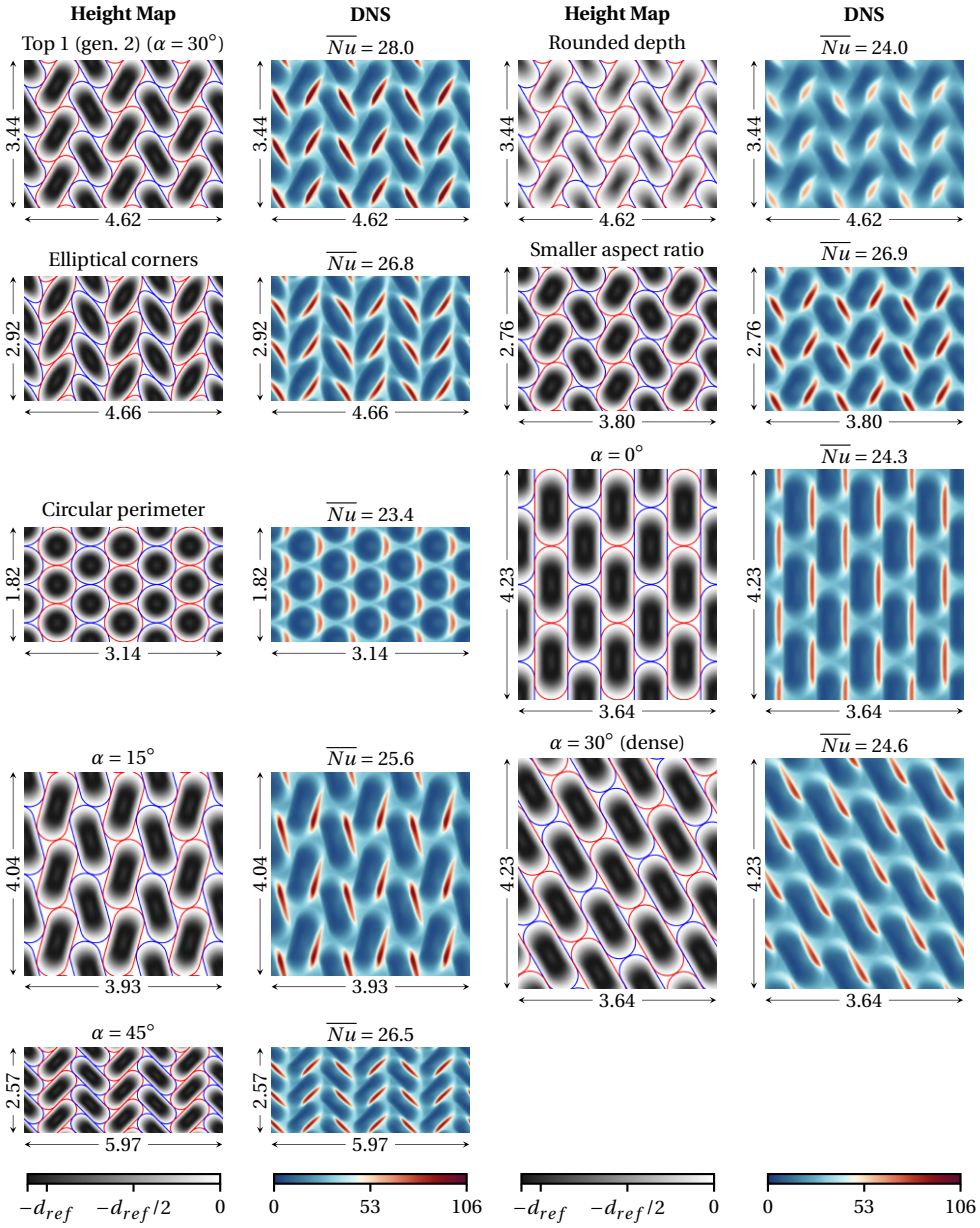


Figure 5.11: Comparison between the optimal dimpled surface found in the second generation of the reinforcement learning loop with other closely related geometries at $Re_\tau = 180$. The bulk flow moves from left to right.

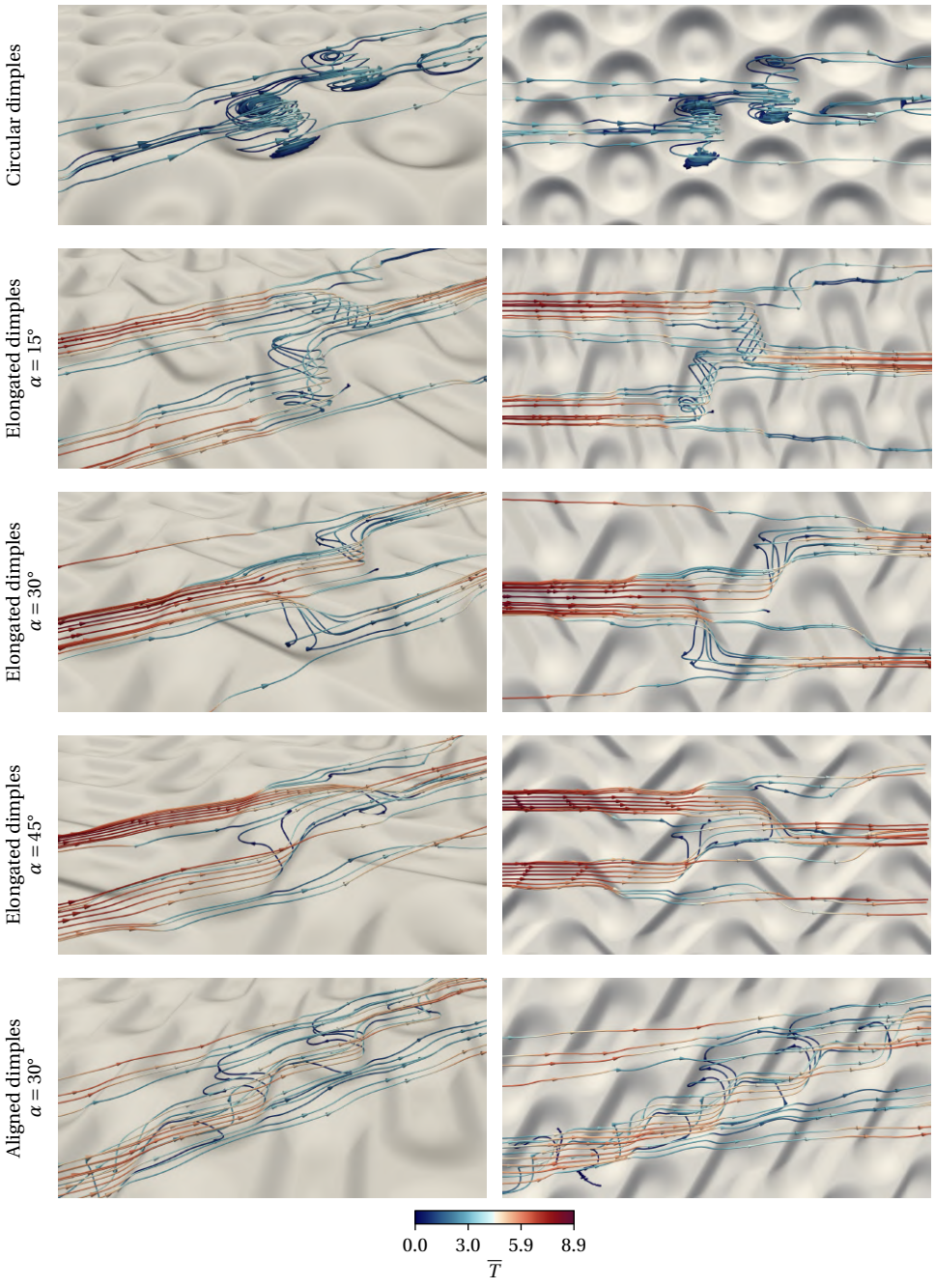


Figure 5.12: Streamlines for the dimples with circular perimeter (left) shown in Fig. 5.11, and the top-ranked dimpled surface (right) obtained in the second generation of the reinforcement learning loop (Figure 5.9). The color-maps for each dimple indicate the averaged temperature ($\overline{T}$) for their respective streamlines.

Figure 5.13: Instantaneous snapshot of the velocity field in the streamwise direction (U^+) for the highest-performing dimpled surface of the second generation of the reinforcement learning study at $Re_\tau = 180$. The slice shown corresponds to the plane at $z = L_z/2$. Please note that $U = U^+$ in our numerical framework. The bulk flow moves from left to right.

5.3.2. OPTIMIZATION FOR DIMPLES SURFACES AT $Re_\tau = 395$

In the previous section, it was shown that machine learning can accurately predict optimized dimpled surface designs for flows at a friction Reynolds number of $Re_\tau = 180$. However, to further demonstrate the ability of the optimization method, we repeated the procedure for flows at $Re_\tau = 395$. To begin this new phase of the study, a new DNS database was generated, considering the randomized dimpled surfaces previously shown in Figure 5.6. Then, the neural network was re-trained using the new DNS database with flows at $Re_\tau = 395$, and the evaluation procedure with millions of combinations was repeated. The top configurations predicted by machine learning are shown in Figure 5.14. Here, it can be seen that the top-ranked dimpled surface resembles the previous optimal design for $Re_\tau = 180$. However, instead of sharper (rounded) corners, elliptical corners are chosen instead. To further investigate this outcome, an additional DNS was performed at $Re_\tau = 395$, but with sharper corners instead. The last row of Figure 5.14 shows a comparison with respect to a dimpled surface with sharper (rounded) corners. From the DNS results, it can be seen that the elliptical corners are indeed beneficial to increase the Nusselt number ($\overline{Nu}$) at $Re_\tau = 395$.

The streamline patterns for DNS cases with both corner effects can be found in Figure 5.15. Here, it can be observed that the elliptical corners create spiral vortices more closely aligned with the dimple geometry, and that most streamlines are ejected together towards the end of each dimple. In contrast, the dimples with sharp (rounded) corners have a more dispersed flow pattern, with the streamlines spilling over the dimples and being ejected at different locations. This implies that the geometry is not closely aligned with the vortices generated, and thus it is natural to expect a higher Nusselt number from the dimples with elliptical corners at $Re_\tau = 395$.

A comparison between the performance metrics for the newly optimized surface at $Re_\tau = 395$ and other relevant cases can be found in Table 5.4. Here, the Nusselt numbers $\overline{Nu}$ tend to be higher than before (Table 5.3), since this quantity grows together Re_b or Re_τ in our physical formulation. However, we highlight that the newly optimized surface (with elliptical corners) has higher values for $Q/W_p/(Q_0/W_{p,0})$ than the previous study at

$Re_\tau = 180$. This indicates that the dimpled surface is operating efficiently, achieving both a high Nusselt number and a good ratio of heat transfer to pumping power. Again, the ratio $\overline{Nu}/\overline{Nu}_0/(C_f/C_{f,0})^{1/3}$ has a favorable value for the optimized surface.

Table 5.4: Thermal performance metrics for the highest performing dimpled surface during the optimization study with $Re_\tau = 395$, and other closely related alternatives. The sub-index "0" refers to a flat-plate channel case at identical bulk Reynolds number (Re_b) as the target configuration.

Case	$\frac{Q/W_p}{Q_0/W_{p,0}}$	$\overline{Nu}$	C_f	$\frac{2St}{C_f}$	$\frac{\overline{Nu}/\overline{Nu}_0}{(C_f/C_{f,0})^{1/3}}$	$\frac{\overline{Nu}/\overline{Nu}_0}{C_f/C_{f,0}}$
Top dimpled surface ($Re_\tau = 395$)	1.78	52.8	0.023	0.88	1.65	0.77
Circular dimples (same Re_b)	1.65	46.8	0.0199	0.9	1.54	0.79
Smooth walls, $Re_\tau \approx 222$ (same Re_b)	1	21.7	0.0073	1.14	1	1
Smooth walls, $Re_\tau = 395$ (same ΔP)	-	36.6	0.0063	1.16	-	-

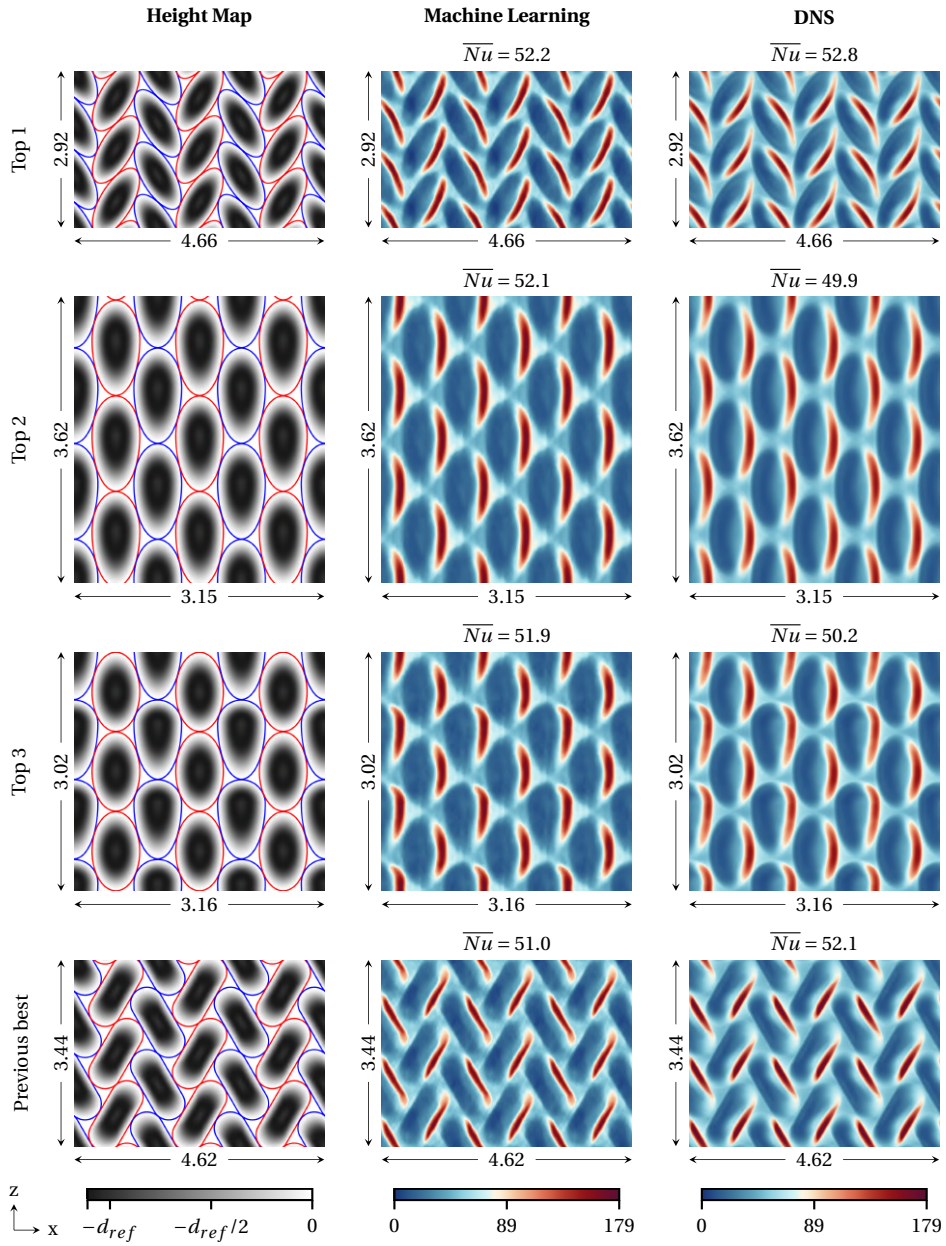


Figure 5.14: Top ranked dimpled surfaces found during the optimization procedure for a friction Reynolds number of $Re_\tau = 395$. The label "previous best" refers to the top-ranked dimpled surface from the second generation of the reinforcement learning study for $Re_\tau = 180$. The bulk flow moves from left to right.

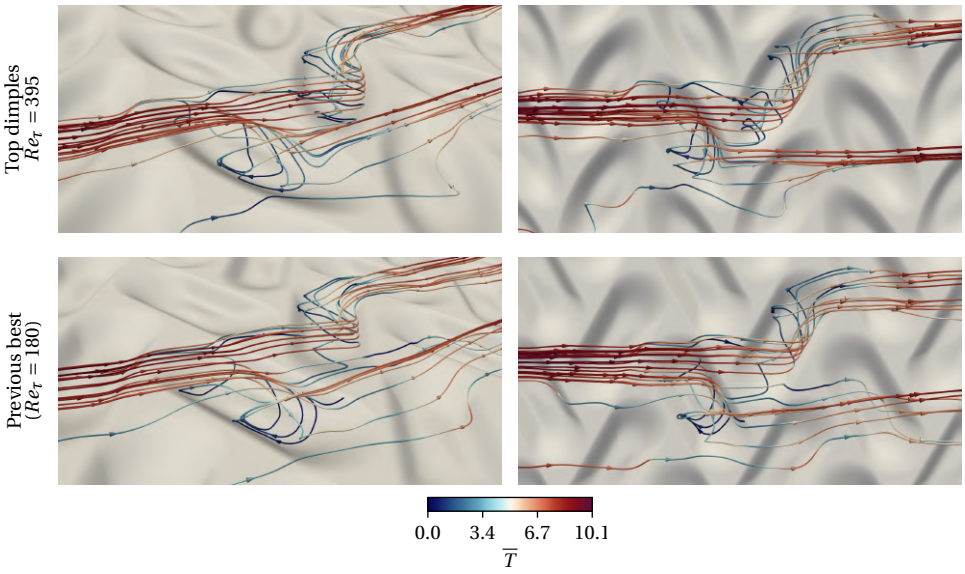


Figure 5.15: Streamlines for the dimples with elliptical corners (top) shown in Fig. 5.14, and the top-ranked dimpled surface (bottom) obtained in the second generation of the reinforcement learning loop (Figure 5.9) at $Re_\tau = 180$. The color-maps for each dimple indicate the averaged temperature ($\overline{T}$) for their respective streamlines.

5.3.3. DISCUSSION OF RESULTS

Regarding the quantitative results presented here, it should be mentioned that the topologies found will only be optimal for the investigated Reynolds numbers. Moreover, the optimized topology was found using Dirichlet boundary conditions. This modeling assumption may not always be correct when dealing with heat transfer equipment. A good example would be a heat exchanger without any multiphase phenomena on either the hot or cold side. However, the study's methodology can be extended to the point where the surface topology is not only optimized on one side, but on both the hot- and cold-side in tandem. In such a case, only the configuration of the DNS solver would need to change to allow for conjugate heat transfer, but the convolutional neural network would still work and thus, the methods in this chapter could be used to optimise topologies that are more relevant to industry. In subsequent studies, the dimple depth should be included in the parametrization of the dimple geometry. Furthermore, the same general methodology can be used for different designs such as transverse bars or riblets, as long as the parametrization is changed accordingly. Finally, to include much higher Reynolds numbers, Large Eddy Simulations could be used instead of Direct Numerical Simulations to extend the applicability of the methods to a wide range of industrial cases.

5.4. CONCLUSIONS

In this study, we present a machine learning framework to optimize rough surfaces for convective heat transfer enhancement. The procedure starts by considering a DNS database with purely random surface designs. Then, a neural network is trained using the existing DNS data, and a new optimal design is predicted within our parameter space. To advance in the reinforcement learning loop, the DNS database is augmented by simulating both the new optimal surface, as well as closely related random variations of this design. The neural network can predict highly optimized dimpled surface designs, starting from the first iteration of the reinforcement learning loop. The rough surfaces identified by machine learning contain elongated dimples with opposite inclination angles, which create a zig-zag pattern for the flow near the walls. This design is highly effective for heat transfer enhancement, and further analysis shows that it is substantially more effective than other plausible alternatives within our parameter space. For instance, smaller dimples with high packing density can have more heat transfer area, yet their Nusselt number is inferior since their shapes do not enhance local heat transfer. Additionally, we have shown that at different Reynolds numbers, the optimal geometry is different as well. This showcases the ability of our machine learning system to prioritize the alignment of roughness elements, and to select other appropriate surface features. Achieving similar results using traditional correlations would be difficult, since standard surface metrics (e.g. skewness) correspond to global averages, which are not sensitive to the exact location of roughness elements. Thus, traditional correlations are not well-posed to predict how the alignment between dimples would enhance heat transfer.

In summary, we conclude that machine learning can be an effective tool to optimize rough surfaces for convective heat transfer enhancement. While typical reinforcement learning problems can require hundreds, if not thousands, of iterations to converge, our system can build highly effective surface designs in only a few iterations. Beyond the machine learning framework, our DNS analysis also shows that elongated dimples with

opposite inclination angles are an interesting alternative to consider in optimization studies, since they produce a large enhancement in the average Nusselt number under equal pressure losses. Moreover, the local Nusselt numbers $Nu(x,z)$ predicted for each configuration serve as an additional verification tool, to check whether the distributions follow physically plausible patterns, or if the mean Nusselt number $\overline{Nu}$ are affected by spurious non-physical oscillations. Further design insights can also be obtained from $Nu(x,z)$ to predict optimal heat transfer patterns, or to understand the sensitivity of the neural network to changes in the dimple geometry.

REFERENCES

Abdus Samad, K.-D. L., & Kim, K.-Y. (2010). Shape optimization of a dimpled channel to enhance heat transfer using a weighted-average surrogate model. *Heat Transfer Engineering*, *31*(13), 1114–1124. https://doi.org/10.1080/01457631003640453

Afanasyev, V., Chudnovsky, Y., Leontiev, A., & Roganov, P. (1993). Turbulent flow friction and heat transfer characteristics for spherical cavities on a flat plate. *Experimental Thermal and Fluid Science*, *7*(1), 1–8. https://doi.org/https://doi.org/10.1016/0894-1777(93)90075-T

Bons, J. (2005). A critical assessment of reynolds analogy for turbine flows. *Journal of Heat Transfer*, *127*(5), 472–485. https://doi.org/10.1115/1.1861919

Busse, A., Lützner, M., & Sandham, N. D. (2015). Direct numerical simulation of turbulent flow over a rough surface based on a surface scan. *Computers & Fluids*, *116*, 129–147. https://doi.org/https://doi.org/10.1016/j.compfluid.2015.04.008

Chen, Y., Chew, Y. T., & Khoo, B. C. (2012). Enhancement of heat transfer in turbulent channel flow over dimpled surface. *International Journal of Heat and Mass Transfer*, *55*(25-26), 8100–8121. https://doi.org/10.1016/j.ijheatmasstransfer.2012.08.043

Cheraghi, M. H., Ameri, M., & Shahabadi, M. (2020). Numerical study on the heat transfer enhancement and pressure drop inside deep dimpled tubes. *International Journal of Heat and Mass Transfer*, *147*, 118845. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2019.118845

Choi, S. M., Kwon, H. G., Bae, H. M., Moon, H. K., & Cho, H. H. (2023). Effects of staggered dimple array under different flow conditions for enhancing cooling performance of solar systems. *Applied Energy*, *342*, 121120. https://doi.org/https://doi.org/10.1016/j.apenergy.2023.121120

Chung, D., Hutchins, N., Schultz, M. P., & Flack, K. A. (2021). Predicting the drag of rough surfaces. *Annual Review of Fluid Mechanics*, *53*(1), 439–471. https://doi.org/10.1146/annurev-fluid-062520-115127

Costa, P. (2025). CaNS Version 3.0: A code for fast, massively-parallel direct numerical simulations (DNS) of canonical flows. https://github.com/CaNS-World/CaNS/releases/tag/v3.0.0

Dagdevir, T., Keklikcioglu, O., & Ozceyhan, V. (2019). Heat transfer performance and flow characteristic in enhanced tube with the trapezoidal dimples. *International*

Communications in Heat and Mass Transfer, *108*, 104299. https://doi.org/https://doi.org/10.1016/j.icheatmasstransfer.2019.104299

Diez, R. (2025). Example of neural network to predict the local nusselt number distributions in turbulent flows past rough surfaces. https://github.com/Rafael10Diez/Machine-Learning-Distributed-Nusselt-Numbers-Efficient

Dipprey, D., & Sabersky, R. (1963). Heat and momentum transfer in smooth and rough tubes at various prandtl numbers. *International Journal of Heat and Mass Transfer*, *6*(5), 329–353. https://doi.org/https://doi.org/10.1016/0017-9310(63)90097-8

Elyyan, M. A., Rozati, A., & Tafti, D. K. (2008). Investigation of dimpled fins for heat transfer enhancement in compact heat exchangers. *International Journal of Heat and Mass Transfer*, *51*(11), 2950–2966. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2007.09.013

Fadlun, E., Verzicco, R., Orlandi, P., & Mohd-Yusof, J. (2000). Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics*, *161*(1), 35–60. https://doi.org/https://doi.org/10.1006/jcph.2000.6484

Forooghi, P., Stripf, M., & Frohnapfel, B. (2018). A systematic study of turbulent heat transfer over rough walls. *International Journal of Heat and Mass Transfer*, *127*, 1157–1168. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2018.08.013

Garai, A., & Murman, S. M. (2021). Stabilization of the adjoint for turbulent flows. *AIAA Journal*, *59*(6), 2001–2013. https://doi.org/10.2514/1.J059998

Isaev, S., Leontiev, A., Chudnovsky, Y., Nikushchenko, D., Popov, I., & Sudakov, A. (2019). Simulation of vortex heat transfer enhancement in the turbulent water flow in the narrow plane-parallel channel with an inclined oval-trench dimple of fixed depth and spot area. *Energies*, *12*(7). https://doi.org/10.3390/en12071296

Isaev, S., Leontiev, A., Gritskevich, M., Nikushchenko, D., Guvernyuk, S., Sudakov, A., Chung, K.-M., Tryaskin, N., Zubin, M., & Sinyavin, A. (2023). Development of energy efficient structured plates with zigzag arrangement of multirow inclined oval trench dimples. *International Journal of Thermal Sciences*, *184*, 107988. https://doi.org/https://doi.org/10.1016/j.ijthermalsci.2022.107988

Jeong, M., Ha, M. Y., & Park, Y. G. (2019). Numerical investigation of heat transfer enhancement in a dimpled cooling channel with different angles of the vortex generator. *International Journal of Heat and Mass Transfer*, *144*, 118644. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2019.118644

Kim, J., Moin, P., & Moser, R. (1987). Turbulence statistics in fully developed channel flow at low reynolds number. *Journal of Fluid Mechanics*, *177*, 133–166. https://doi.org/10.1017/S0022112087000892

Kim, K.-H., Kang, J.-H., Pan, X., & Choi, J.-I. (2023). Pascal_tcs: A versatile solver for large-scale turbulent convective heat transfer problems with temperature-dependent fluid properties. *Computer Physics Communications*, *290*, 108779. https://doi.org/https://doi.org/10.1016/j.cpc.2023.108779

Kumar, A., Maithani, R., & Suri, A. R. S. (2017). Numerical and experimental investigation of enhancement of heat transfer in dimpled rib heat exchanger tube. *Heat and Mass Transfer*, *53*(12), 3501–3516. https://doi.org/10.1007/s00231-017-2080-x

Leonardi, S., Orlandi, P., Djenidi, L., & Antonia, R. A. (2015). Heat transfer in a turbulent channel flow with square bars or circular rods on one wall. *Journal of Fluid Mechanics*, *776*, 512–530. https://doi.org/10.1017/jfm.2015.344

Li, M., Khan, T. S., Al-Hajri, E., & Ayub, Z. H. (2016). Single phase heat transfer and pressure drop analysis of a dimpled enhanced tube. *Applied Thermal Engineering*, *101*, 38–46. https://doi.org/10.1016/j.applthermaleng.2016.03.042

Ligrani, P. M., Oliveira, M. M., & Blaskovich, T. (2003). Comparison of heat transfer augmentation techniques. *AIAA Journal*, *41*(3), 337–362. https://doi.org/10.2514/2.1964

Ligrani, P. (2013). Heat transfer augmentation technologies for internal cooling of turbine components of gas turbine engines. *International Journal of Rotating Machinery*, *2013*, 275653. https://doi.org/10.1155/2013/275653

Orlandi, P., & Leonardi, S. (2006). DNS of turbulent channel flows with two- and three-dimensional roughness. *Journal of Turbulence*, *7*, N73. https://doi.org/10.1080/14685240600827526

Peeters, J., & Sandham, N. (2019). Turbulent heat transfer in channels with irregular roughness. *International Journal of Heat and Mass Transfer*, *138*, 454–467. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2019.04.013

Rao, Y., Feng, Y., Li, B., & Weigand, B. (2015). Experimental and Numerical Study of Heat Transfer and Flow Friction in Channels With Dimples of Different Shapes. *Journal of Heat Transfer*, *137*(3), 031901. https://doi.org/10.1115/1.4029036

Rao, Y., Li, B., & Feng, Y. (2015). Heat transfer of turbulent flow over surfaces with spherical dimples and teardrop dimples. *Experimental Thermal and Fluid Science*, *61*, 201–209. https://doi.org/https://doi.org/10.1016/j.expthermflusci.2014.10.030

Rao, Y., Zhang, P., Xu, Y., & Ke, H. (2020). Experimental study and numerical analysis of heat transfer enhancement and turbulent flow over shallowly dimpled channel surfaces. *International Journal of Heat and Mass Transfer*, *160*, 120195. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2020.120195

Rashidi, S., Hormozi, F., Sundén, B., & Mahian, O. (2019). Energy saving in thermal energy systems using dimpled surface technology – a review on mechanisms and applications. *Applied Energy*, *250*, 1491–1547. https://doi.org/https://doi.org/10.1016/j.apenergy.2019.04.168

Rouhi, A., Endrikat, S., Modesti, D., Sandberg, R. D., Oda, T., Tanimoto, K., Hutchins, N., & Chung, D. (2022). Riblet-generated flow mechanisms that lead to local breaking of reynolds analogy. *Journal of Fluid Mechanics*, *951*, A45. https://doi.org/10.1017/jfm.2022.880

Salvadore, F., Bernardini, M., & Botti, M. (2013). Gpu accelerated flow solver for direct numerical simulation of turbulent flows. *Journal of Computational Physics*, *235*, 129–142. https://doi.org/https://doi.org/10.1016/j.jcp.2012.10.012

Sanhueza, R. D., Akkerman, I., & Peeters, J. W. (2023). Machine learning for the prediction of the local skin friction factors and nusselt numbers in turbulent flows past rough surfaces. *International Journal of Heat and Fluid Flow*, *103*, 109204. https://doi.org/https://doi.org/10.1016/j.ijheatfluidflow.2023.109204

Sanhueza, R. D., Peeters, J., & Costa, P. (2025). A pencil-distributed finite-difference solver for extreme-scale calculations of turbulent wall flows at high reynolds number. https://arxiv.org/abs/2502.06296

Sanhueza, R. D., & Peeters, J. W. (2025). Data-driven optimization of rough surfaces for convective heat transfer enhancement. *International Journal of Heat and Mass Transfer*, *251*, 127313. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2025.127313

Thakkar, M., Busse, A., & Sandham, N. (2018). Direct numerical simulation of turbulent channel flow over a surrogate for nikuradse-type roughness. *Journal of Fluid Mechanics*, *837*. https://doi.org/10.1017/jfm.2017.873

Thakkar, M., Busse, A., & Sandham, N. (2017). Surface correlations of hydrodynamic drag for transitionally rough engineering surfaces. *Journal of Turbulence*, *18*(2), 138–169. https://doi.org/10.1080/14685248.2016.1258119

Turnow, J., Kornev, N., Zhdanov, V., & Hassel, E. (2012). Flow structures and heat transfer on dimples in a staggered arrangement. *International Journal of Heat and Fluid Flow*, *35*, 168–175. https://doi.org/10.1016/j.ijheatfluidflow.2012.01.002

Walsh, M. J., & Weinstein, L. M. (1979). Drag and heat-transfer characteristics of small longitudinally ribbed surfaces. *AIAA Journal*, *17*(7), 770–771. https://doi.org/10.2514/3.61216

Xia, H., Tang, G., Shi, Y., & Tao, W. (2014). Simulation of heat transfer enhancement by longitudinal vortex generators in dimple heat exchangers. *Energy*, *74*, 27–36. https://doi.org/https://doi.org/10.1016/j.energy.2014.02.075

Xie, S., Liang, Z., Zhang, L., & Wang, Y. (2018). A numerical study on heat transfer enhancement and flow structure in enhanced tube with cross ellipsoidal dimples. *International Journal of Heat and Mass Transfer*, *125*(100), 434–444. https://doi.org/10.1016/j.ijheatmasstransfer.2018.04.106

Yang, M., Kang, J.-H., Kim, K.-H., Kwon, O.-K., & Choi, J.-I. (2023). Pascal_tdma 2.0: A multi-gpu-based library for solving massive tridiagonal systems. *Computer Physics Communications*, *290*, 108785. https://doi.org/https://doi.org/10.1016/j.cpc.2023.108785

Zhengyi Wang, K. S. Y., & Khoo, B. C. (2006). DNS of low reynolds number turbulent flows in dimpled channels. *Journal of Turbulence*, *7*, N37. https://doi.org/10.1080/14685240600595735

6

CONCLUSIONS AND PERSPECTIVES

The work performed in this thesis addressed multiple challenges in the field of machine learning for fluid mechanics. The topics included: improving GPU-accelerated DNS solvers for extreme-scale simulations, developing accurate machine learning models for variable-property flows, predicting the behavior of turbulent flows past rough surfaces, and optimizing dimpled surface configurations for convective heat transfer enhancement. The combination of machine learning and GPU-accelerated flow solvers is seen as a promising technique. Experience shows that GPUs can quickly generate large turbulent flow databases, with a large variety of complex physical effects beyond the scope of (simplified) traditional models. Machine learning models can then be trained using large flow databases, and complex physical effects can be fully taken into account to make accurate predictions.

GPU-ACCELERATED DNS SOLVERS FOR EXTREME-SCALE SIMULATIONS

This work addressed two important topics regarding GPU-accelerated DNS solvers. First, a parallel tridiagonal solver (P-TDMA) with reduced MPI communication footprint for extreme-scale GPU simulations was implemented. Then, a new cross-platform communication library, named *diezDecomp*, was created for DNS studies in either AMD or NVIDIA GPUs. This library supports highly complex data transfer operations, maintaining high performance at extreme-scales. The results show that the parallel tridiagonal solver is highly effective in extreme-scale simulations with up to 1,024 GPUs/GCDs in both the Leonardo and LUMI supercomputers. More concretely, the P-TDMA algorithm enables the DNS solver to use a 2-D pencil decomposition, while retaining the same levels of speed observed in high-performing 1-D slabs decompositions. Additionally, while P-TDMA algorithms are often conceived as handling large-scale simulations by adding more 1-D slabs along the vertical direction, the results clearly showed that it is better to minimize the number of vertical partitions, and to add more lateral divisions for extreme-scale simulations. This insight is consistent with theoretical analysis, since DNS solvers with more lateral partitions have lower interface area in the vertical direction, and thus the MPI communication footprint of the P-TDMA algorithm is effective halved when the

number of GPUs is duplicated. In contrast, P-TDMA algorithms with 1-D slabs have a constant interface area, and their communication footprint and running times do not necessarily improve by adding more GPUs. Still, P-TDMA algorithms with 1-D slabs can be highly effective for medium-sized DNS simulations when the number of grid points in the vertical direction is very large for every MPI task. In general, the optimal DNS solver configuration is a hardware-dependent choice that is also influenced by the DNS grid size and the (non-uniform) communication bandwidth observed between intra-node and multi-node data transfers. The P-TDMA algorithm with a 2-D pencil decomposition and a minimum number of vertical partitions always offers good performance, yet in some cases, either the P-TDMA algorithm with 1-D slabs or the full-transpose algorithm with 1-D slabs can be (slightly) more effective.

The new cross-platform communication library for NVIDIA or AMD GPU-based supercomputers, named *diezDecomp*, proved to be highly effective at performing data transfer in large-scale simulations. The working principle of this library is to intersect the x/y/z bounds of all MPI processes and to schedule data transfers. Therefore, highly complex communication patterns are natively supported. Thanks to this flexibility, a new type of transposes in the x-z direction was added to the DNS solver, which enable one-step data transfers between x-aligned pencils and any type of z-aligned pencil arrangement. Due to technical reasons, x-z transposes are beneficial when solving for 1-D implicit diffusion equations inside DNS solvers, yet they require a complex communication pattern that is not possible to achieve with operations resembling all-to-all collective operations. The results showed that x-to-z transposes can substantially reduce the total running times for DNS solvers using 2-D pencil decomposition schemes with a high number of partitions in the z-direction.

In summary, the previous advances in GPU-based DNS solvers are important, since they enable the generation of high-fidelity data (regarding complex flows) in short periods of time. This works well in combination with machine learning, which can be used to quickly build new low-cost models for engineering purposes.

MACHINE LEARNING FOR VARIABLE PROPERTY FLOWS

In the field of machine learning for variable property flows, the analysis focused on improving the behavior of existing RANS turbulence models, showing significant errors due to the presence of strong property gradients. It is well-documented in the literature that CFD solvers naively enabling property variations (density, viscosity, conductivity, etc.) can still have large errors due to variable-property effects, such as density gradients. The machine learning framework is based on the FIML (Field Inversion Machine Learning) technique proposed by Parish & Duraisamy (2016) to account for strong property variations. Modelling variable-property flows has additional challenges, such as not knowing beforehand the thermophysical properties of the flows studied, since they vary strongly with the temperature.

To solve this challenge, a feedback loop was added, where the data-augmented RANS predictions are used to estimate the fluid temperature and to update the local thermophysical properties. These changes are propagated even through the input features given to the machine learning system. Additionally, to avoid adding non-physical corrections to RANS turbulence models, a weighted relaxation factor is proposed, which can effectively

filter spurious oscillations in real-time by solving a single scalar equation. The final results showed that machine learning system was highly effective at improving CFD predictions, and it even had good performance when it was forced to extrapolate to a higher Reynolds number.

Turbulent flows past rough surfaces

Another challenge was to predict the thermal and hydrodynamic behavior of turbulent flows past rough surfaces. This is a complex task, as it requires the evaluation of many complex features found in rough surfaces. In general, traditional surface metrics over-simplify the characteristics of a rough surface, and they are not sensitive to the original location of the roughness elements being processed. Due to the previous challenges, it was decided to use a convolutional neural network architecture, which is capable of (independently) scanning the height map of a rough surface, and to make predictions for the local skin friction factors and Nusselt numbers. This approach is well-posed numerically, since predicting detailed 2-D maps for the entire surface creates a one-to-one ratio between the height map of the rough surface (input features) and the target predictions (local 2-D skin friction or Nusselt numbers). This is beneficial for the training procedure of the neural network, and it allows the machine learning system to extract more physical information from smaller databases. Additionally, an advanced system is proposed to recycle the intermediate results of all convolutional layers, such that 2-D surface maps can be predicted with linear time complexity, instead of quadratic complexity as in naive approaches.

The results show that machine learning can accurately predict the skin friction factors and Nusselt number distributions for rough surfaces in the dataset. Yet, more generally, the results are exceptionally accurate for the thermal predictions, regarding the Nusselt numbers. This is attributed to the relatively simpler nature of the scalar transport equation for the temperature, which is not influenced by complex effects, such as the buildup of pressure drag near roughness elements that influences the skin friction factor predictions.

Data-driven optimization of rough surfaces for convective heat transfer enhancement

Based on the success of the optimized convolutional neural network architecture for thermal predictions, a data-driven optimization system was developed for convective heat transfer enhancement in dimpled-surfaces. The system is based on a (basic) reinforcement learning approach. First, a database is created with random dimpled-surface examples, which are used to train a neural network to make heat transfer predictions. Then, millions of dimpled surface designs are evaluated, seeking the optimal heat transfer configurations, and a new optimized surface is predicted. The results show that machine learning is very accurate at predicting highly optimized surface designs starting from low-quality training data. A database with only 20 randomized dimpled-surfaces displaying sub-optimal Nusselt numbers was enough to enable machine learning to extract meaningful physical patterns and predict that cross-aligned elongated dimple elements would yield the most heat transfer. This prediction was found to be highly accurate, and it is indeed one of the highest-performing dimpled-surfaces in the current parameter space. Physically speaking, cross-aligned elliptical dimples deviate the flow in coordinated zig-zag patterns, which enhance the heat transfer downstream instead of blocking the flow.

This solves a problem typically occurring in circular dimples, where the main flow hovers over the top of the dimples, and only secondary recirculation vortices are found inside. This recirculation causes a region with low Nusselt numbers.

After extending the dataset of the machine learning system, to complete a second reinforcement learning iteration, it was found that machine learning again predicted improved dimpled surface parameters. This time machine learning identified the correct dimensions for multiple dimple features, such as the corner effects, inclination angle, etc. Multiple DNS runs were performed to test other plausible dimpled-surface configurations, yet it was found that machine learning had identified the best parameters among such alternatives. The best dimpled surface found using reinforcement learning has a Nusselt number 53% higher than a flat plate configuration. Therefore, machine learning is a useful tool both to predict turbulence flow quantities, and to optimize engineering equipment for enhanced thermal performance.

FINAL REMARKS

GPU-accelerated DNS solvers are creating a new breakthrough in turbulent flow simulations, where it is now possible to simulate highly complex flows in short time periods. This creates significant opportunities to study complex physical effects, and to use machine learning to extract patterns from rich databases. The work performed has shown that machine learning is an effective tool to make accurate predictions for complex flows with strong gradients in their thermophysical properties, or flows that are influenced by the presence of rough surfaces. Our work optimizing dimpled surfaces for heat transfer enhancement also shows how machine learning can enable the direct optimization of engineering equipment for improved thermal performance. Therefore, machine learning is a promising tool in the field of fluid mechanics.

PERSPECTIVES

In broad terms, machine learning can be applied in fluid mechanics to solve important challenges, such as designing complex machinery, building data-driven CFD solvers, or generating low-cost (fast) turbulent flow predictions for process control and optimization. Within this context, several guidelines should be considered. First, it is essential to create a well-defined parameter space for each machine learning application, where all degrees of freedom can be represented by scalar quantities within closed intervals. This enables the identification of relevant training and validation cases for the study. Even if there are millions of combinations in the parameter space, our work showcased how an optimized dimpled surface (with 53% higher Nusselt number) could be identified after training a neural network with 20 randomly chosen simulations. Each simulation with a refined grid took less than 20 hours using a single GPU, and training the neural network again took a similar amount of time. Therefore, it is possible to use machine learning to identify optimal solutions in less than a week, among millions of alternatives. Using random training cases tends to be effective, because each sample contains non-linear interactions between many factors, and thus machine learning is always being trained to work with non-trivial cases. In constrast, choosing edge cases as training data can be useful, if they correspond to optimized configurations, but otherwise it may not be as effective. To regularize machine learning systems, a technique we found to be effective was to systematically reduce the size of the machine learning system, while retaining

a similar training and validation accuracy. This can be seen as an application of the Elbow Method, where redundant parameters that may cause over-fitting are eliminated. Additionally, the ratio between the number of trainable parameters and the size of the dataset is improved. For enhanced accuracy, classical L1 or L2 regularization can be used in combination with the experiments to reduce the size of the machine learning system. Finally, regarding the usage of standalone machine learning systems, or data-augmented physical models, several trade-offs should be considered. On one hand, physical models may only need fine-tuning for isolated constants, and physical laws (e.g., mass or momentum conservation) are automatically fulfilled. However, some frameworks over-simplify the flow physics, and they cannot replicate the experimental data with high-fidelity. Moreover, physical models can create numerical stability or scalability issues; especially when optimizing parameters in non-linear PDEs. Thus, using standalone machine learning systems can be feasible, unless a physical framework provides strong benefits.

In summary, machine learning has great potential in fluid mechanics, yet the right mathematical techniques are problem-specific and should be chosen carefully.

ACKNOWLEDGEMENTS

I would like to thank my supervisors dr. ir. Jurriaan Peeters, dr. ir. Ido Akkerman and prof. dr. ir. Bendiks Jan Boersma for their guidance during my thesis, as well as the opportunity to work in the fields of machine learning for fluid mechanics and GPU-accelerated DNS solvers. I am grateful to dr. Pedro Costa for his extensive support and mentoring during the development of HPC software for extreme-scale GPU simulations in supercomputers. I would further like to thank dr. ir. Jurriaan Peeters for his frequent help and training in topics such as heat transfer simulations, DNS of flows with rough surfaces, or Fortran programming. I appreciate the guidance of prof. dr. ir. Rene Pecnik during our academic collaboration developing machine learning models for variable-property flows, as well as the opportunity to work on the Snellius supercomputer.

I acknowledge the support of EuroHPC, SURF and NWO giving me access to computing time on the Snellius, LUMI and Leonardo supercomputers. I am also grateful to dr. Josh Romero and dr. Massimiliano Fatica for our insightful discussion about data communication in GPU systems. I am thankful to profs. Rene Pecnik, Jurriaan Peeters, Sergio Pirozzoli and Pedro Costa for writing research proposals applying for computing time. The training courses and advice of the LUMI support staff were important during the development of GPU-accelerated simulations in AMD-based supercomputers.

I thank the PhD thesis committee members for their time and effort assessing this thesis report. I appreciate the help of dr. Ashish Patel, dr. Andrew Trettel and dr. Johan Larsson for the access to their variable-property DNS data.

I am grateful to my friends and colleagues for their constant support and motivation throughout my studies, as well as the organization of activities. Finally, I would like to thank my family for their unconditional love and support throughout my life.

A

APPENDIXES CHAPTER 2

A.1. GPU IMPLEMENTATION OF FOURIER-BASED TRANSFORM

While a real-to-complex Fourier transform of a signal $\mathbf{x}$ with n numbers has $n/2+1$ complex numbers, the imaginary parts of the first and last elements (for even n) are zero. Let us consider the output of the real-to-complex Fourier transform of

$$\mathbf{x} = \begin{bmatrix} x_0 & x_1 & \dots & x_{n-1} \end{bmatrix}, \tag{A.1}$$

given by

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \tilde{x}_0^r & \tilde{x}_0^i & \tilde{x}_1^r & \tilde{x}_1^i & \dots & \tilde{x}_{\lfloor n/2+1 \rfloor}^r & \tilde{x}_{\lfloor n/2+1 \rfloor}^i \end{bmatrix}; \tag{A.2}$$

$\tilde{\boldsymbol{x}}$ has $\lfloor n/2+1 \rfloor$ elements, with $\lfloor\rfloor$ denoting the integer floor operation. Since each complex number is represented by two real ones, $\tilde{\boldsymbol{x}}$ is represented by $(2\ \lfloor n/2+1 \rfloor)$ real numbers, with $\tilde{x}_0^i = 0$, and $\tilde{x}_{\lfloor n/2+1 \rfloor}^i = 0$ for even n. Hence, the real-to-complex transform can be uniquely represented by a set of n numbers. This property is explored in several FFT packages (e.g., *FFTPACK*, and the *half-complex* format of FFTW used in the CaNS code for CPU-based runs). Unfortunately, popular GPU-based FFT libraries like *cuFFT*, *hipFFT*, or *MKL* do not support this format ("cuFFT API Reference", 2024; "hipFFT documentation", 2024; "Intel oneAPI Math Kernel Library (oneMKL)", 2024).

Representing the output of the real-to-complex transforms in arrays of size n is desirable, as it allows us to handle the output of a real-to-complex transform in the same manner as a real-to-real transform, greatly simplifying the implementation of different transform types in the Poisson solver. Hence, since the first GPU version of the CaNS code (Costa et al., 2021), $\tilde{\boldsymbol{x}}$ is packed in the following format:

$$\tilde{\boldsymbol{x}}' = \begin{cases} \tilde{x}_0^r \quad \tilde{x}_0^i \quad \cdots \quad \tilde{x}_{\lfloor n/2+1 \rfloor}^r \quad \tilde{x}_{\lfloor n/2+1 \rfloor}^i & \text{if } n \text{ is even,} \\ \tilde{x}_0^r \quad \tilde{x}_0^i \quad \cdots \quad \tilde{x}_{\lfloor n/2+1 \rfloor}^r \quad \tilde{x}_{\lfloor n/2+1 \rfloor}^i & \text{if } n \text{ is odd.} \end{cases} \tag{A.3}$$

It is easily seen that both cases have n real elements. This operation has $\mathcal{O}(1)$ time complexity, while re-arranging the signal such as $[\tilde{x}_0^r, \ldots, \tilde{x}_1^i, \ldots]$, would have $\mathcal{O}(n)$ complexity. With this cheaper re-arrangement of the arrays, the Fourier eigenvalues λ_i and λ_j in eq. (2.12) must be consistently re-ordered to comply with this format. This is an inexpensive operation that is performed during the initialization of the Poisson/Helmholtz solver.

Finally, the reciprocate unpacking operations are done for performing the inverse complex-to-real transform to have an input array with $\lfloor n/2+1 \rfloor$ elements, resulting in an output signal with n-elements in the correct order.

A.2. PERFORMANCE GAINS FROM DIRECT $x \rightarrow z$ TRANSPOSES WITH IMPLICIT 1D DIFFUSION

When the GPU profiling results from Figure 2.7 are analyzed, it can be noticed that the implicit 1D diffusion solver performs two consecutive transposes in the x-y and y-z directions, which will be denoted as $x \rightarrow y \rightarrow z$ in this section. This is suboptimal. Ideally, the cyclic reduction process should be performed using the original x-aligned pencils for the velocity components $u/v/w$, and then direct $x \rightarrow z$ transposes should be used before solving the reduced systems of tridiagonal equations.

To better understand the benefits of performing a direct $x \rightarrow z$ transpose instead of two consecutive transposes, we used the *diezDecomp* library created for the LUMI porting effort. The implementation intersects the $x/y/z$ bounds of different MPI tasks, with no strong restrictions, and thus it is trivial to implement any variant of $x \rightarrow z$ transpose. This allowed for an implementation of this more complex communication operation with minimal changes in the DNS code.

While avoiding $x \rightarrow y \rightarrow z$ transposes has a small impact in the GPU profiling results shown in Figure 2.7, we identified other DNS cases where the impact of this transpose sequence was much higher. For instance, the DNS cases with $(p_y \times p_z) = (8 \times 128)$ have a much higher MPI workload for the parallel tridiagonal solver (due to the reduced size of p_y), and thus they benefit more from removing $x \rightarrow y \rightarrow z$ transposes. In Figure A.1, the results of the scalability tests using $x \rightarrow z$ transposes are presented for a 2D pencil decomposition with $p_z = 128$ vertical partitions. The configuration $p_z = 128$ was chosen, since its parallel tridiagonal solver works with larger arrays and the unnecessary $x \rightarrow y$ transposes have a significant impact in the results previously shown in Figure 2.4. In the subplot (a), it can be seen that the running times for the implicit 1D diffusion solver are 55% slower when successive x-y and y-z transposes are used. The scalability chart at the right reveals that the system with $x \rightarrow z$ transposes is more efficient in large-scale simulations, reducing the running times of the entire DNS solver by 18% for the P-TDMA algorithm with 1 024 GCDs.

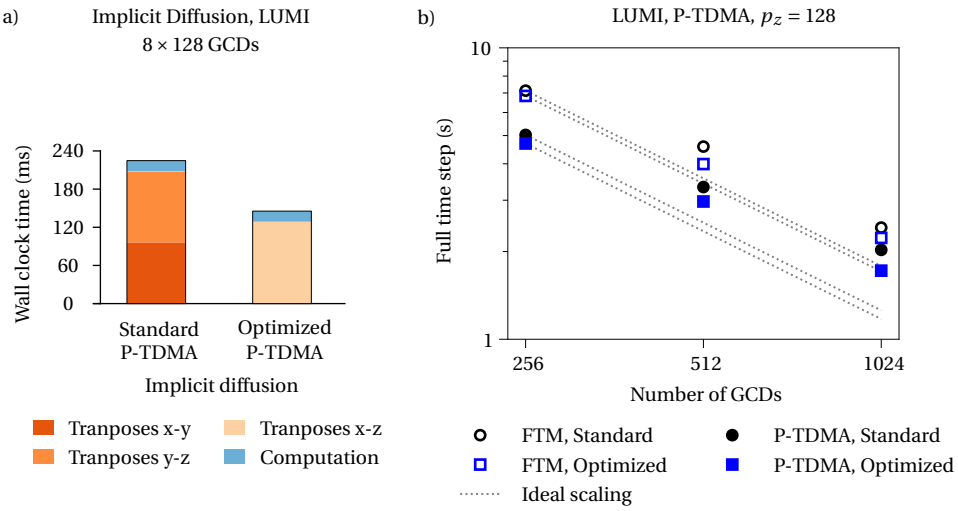


Figure A.1: Comparison between the standard P-TDMA approach and the optimized version using direct $x \rightarrow z$ transpose operations for implicit 1D diffusion (subplot a) using 8×128 GCDs and a grid size of $(N_x \times N_y \times N_z) = (7168 \times 7168 \times 1594)$. The subplot (b) corresponds to a strong scalability test for the entire DNS solver using the same grid size, but a different number of GCDs in the LUMI supercomputer.

REFERENCES

Costa, P., Phillips, E., Brandt, L., & Fatica, M. (2021). Gpu acceleration of cans for massively-parallel direct numerical simulations of canonical fluid flows [Development and Application of Open-source Software for Problems with Numerical PDEs]. *Computers & Mathematics with Applications, 81*, 502–511. https://doi.org/https://doi.org/10.1016/j.camwa.2020.01.002

cuFFT API reference. (2024). https://docs.nvidia.com/cuda/cufft

hipFFT documentation. (2024). https://rocm.docs.amd.com/projects/hipFFT/en/latest

Intel oneAPI Math Kernel Library (oneMKL). (2024). https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html

B

APPENDIXES CHAPTER 4

B.1. TIME AND SPACE COMPLEXITY OPTIMIZATION

During the current study, one of the main challenges encountered while using traditional convolutional neural networks is that their architecture has been adapted to reduce an input image until a single prediction point is obtained, and to discard the results of all intermediate convolutional layers. This process is highly inefficient, as it can be observed in Figure B.1. Here, an example is presented regarding the sub-images required to perform predictions for the local skin friction values $C_f(x,z)$ or Nusselt numbers $Nu(x,z)$ associated with two neighboring data points P_1 and P_2 inside a rough surface. The large overlapping area highlighted in Figure B.1 corresponds to the shared portion of both sub-images which must be scanned redundantly once predictions are required for a second data point P_2. From a mathematical perspective, this implies that in order to obtain predictions for the local skin friction values $C_f(x,z)$ or Nusselt numbers $Nu(x,z)$ associated to a rough surface discretized using images with $N_x \times N_z$ pixels, the total numbers of pixels to be processed by the deep learning architecture is proportional to $\mathcal{O}\left(N_x^2 N_z^2\right)$. In practical terms, a traditional convolutional neural network replicating the behavior of the deep learning systems presented in Tables 4.3 or 4.4 would require processing a total of 365,148,000 pixels for each DNS case considered.

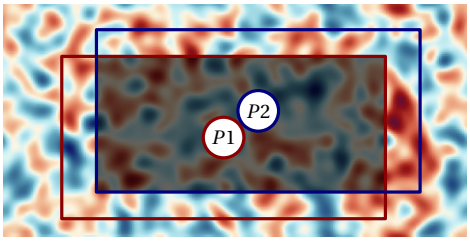


Figure B.1: Example regarding two prediction points $P1$ and $P2$ obtained as output from a traditional convolutional neural network. The shaded area corresponds to the shared portion of the input images passed to the convolutional neural network, which must be reprocessed again.

However, an efficient alternative can be developed to solve the previous problem, and to reduce the total number of pixels to be processed from quadratic complexity $\mathcal{O}\left(N_x^2 N_z^2\right)$ to linear complexity $\mathcal{O}\left(N_x N_z\right)$ with respect to the total image size $N_x N_z$. The approach is based on the idea of sharing the results of all intermediate convolutional layers by using an advanced dilation system that replaces the reduction strides found in traditional convolutional neural networks by a unified matrix system. A schematic representation of this approach can be found in Figure B.2. Here, it can be noted that, instead of reducing images using convolutional strides, the efficient deep learning system proposed uses increased levels of dilation to replicate the effect of using reduction strides. The main benefit of this approach is that neighboring data points, such as P_1 and P_2 back in Figure B.1, can share the results for the intermediate convolutional layers, and all the predictions for the local skin friction values $C_f(x,z)$ or Nusselt numbers $Nu(x,z)$ associated to a rough surface can be obtained after only one pass through the neural network. As a result, the total number of pixels to be processed for each DNS case is reduced from 365,148,000 pixels into only 39,200 pixels, which is equivalent to a reduction factor of 9,315. Moreover, the running times required to train deep learning systems can be reduced from months of GPU-time into only a few hours. The optimized architecture described in Figure B.2 is highly compatible with GPU's, since most machine learning frameworks contain efficient implementations of the dilation operator for convolutions.

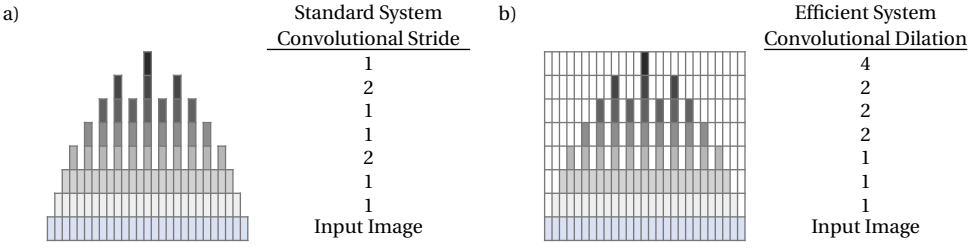


Figure B.2: Comparison in 1-D between a standard convolutional neural network and the efficient computer vision system developed.

B.2. AVERAGED FLOW QUANTITIES AND SURFACE METRICS FOR THE DNS DATABASE

In this Appendix, subplots are presented regarding the distribution of the flow quantities reported in Table 4.1, as well as the value of traditional surface metrics for our DNS database. For improved transparency, the results of Figure B.3 are split for rough surfaces belonging to Categories I and II of the DNS database. As mentioned in Section 4.2.4, all the rough surfaces found in our DNS database have a constant root-mean-squared height of 0.0358.

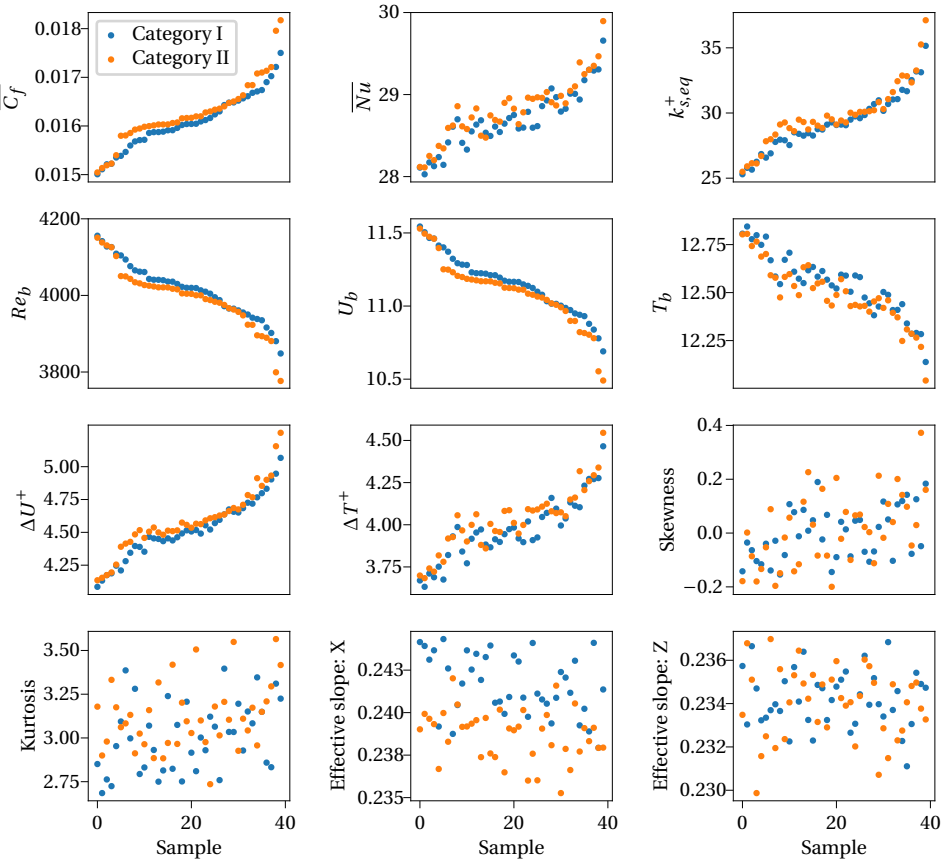


Figure B.3: Distribution of averaged flow quantities reported in Table 4.1 for our DNS database, as well as other traditional surface metrics. For improved readability, all values shown for Categories I and II are sorted in ascending order according to their global skin friction factors $\overline{C_f}$. All the rough surfaces shown in the subplots have a root-mean-squared height of 0.0358.

C

APPENDIXES CHAPTER 5

C.1. VALIDATION OF THE TURBULENT FLOW SOLVER

In order to validate the new GPU-based DNS solver, a comparison was first performed with respect to the channel flow with a grit-blasted surface simulated by (Busse et al., 2015; Peeters & Sandham, 2019; Thakkar et al., 2018). This DNS case uses a friction Reynolds number of $Re_\tau = 180$ and a Prandtl number equal to unity. The grid size considered was $(N_x \times N_y \times N_z) = (560 \times 280 \times 280)$ in each Cartesian direction, whereas the domain size is $(L_x \times L_y \times L_z) = (5.63 \times 2 \times 2.815)$. These settings are equivalent to a dimensionless grid size of $\Delta x^+ = \Delta z^+ = 1.81$ in the streamwise and spanwise directions, respectively. In the wall normal direction, a constant value of $\Delta y^+ = 0.65$ is kept near the walls. The validation data is plotted in Figure C.1. Here, a comparison is performed with respect to the DNS data of (Busse et al., 2015; Peeters & Sandham, 2019). As it can be seen in the sub-plots, the new GPU-based DNS solver closely matches the turbulence statistics reported, and the associated changes in the velocity (ΔU^+) and temperature (ΔT^+) profiles.

To further validate the accuracy of our DNS solver, an additional DNS case was run replicating the work of (Maaß & Schumann, 1996). This case corresponds to a turbulent flow past a wavy surface. The results of the benchmark can be found in Figure C.2. Here, it can be observed that the velocity profiles near the wavy surface are extremely close to the reference data. This is a strong validation of our implementation for the immersed boundary method, since the DNS case of (Maaß & Schumann, 1996) operates at a relatively low Reynolds number with effects like flow separation.

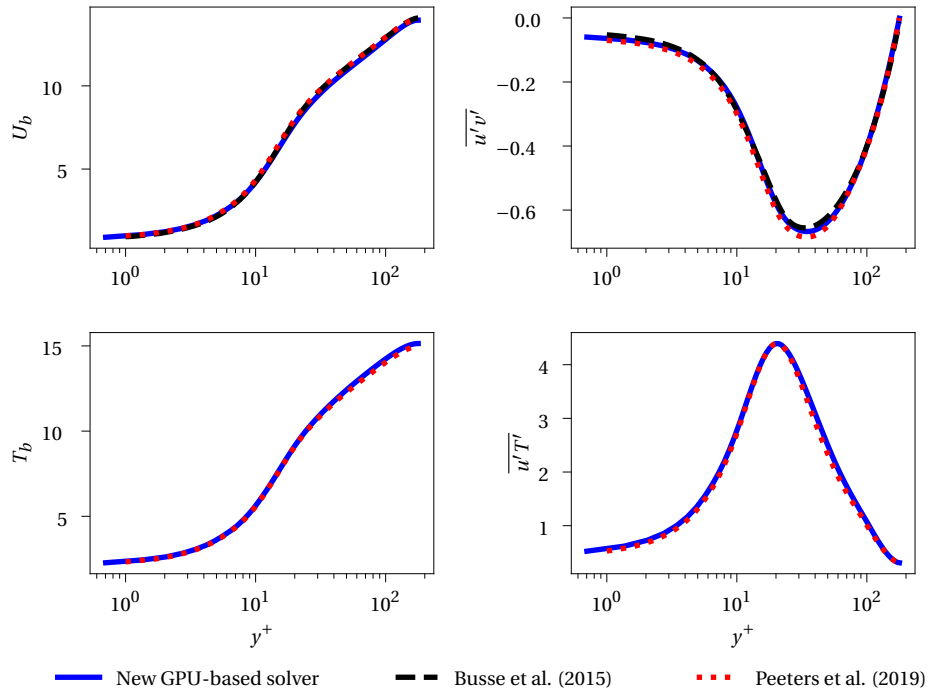


Figure C.1: Comparison between the turbulence statistics for the new GPU-based DNS solver and existing literature (Busse et al., 2015; Peeters & Sandham, 2019; Thakkar et al., 2018).

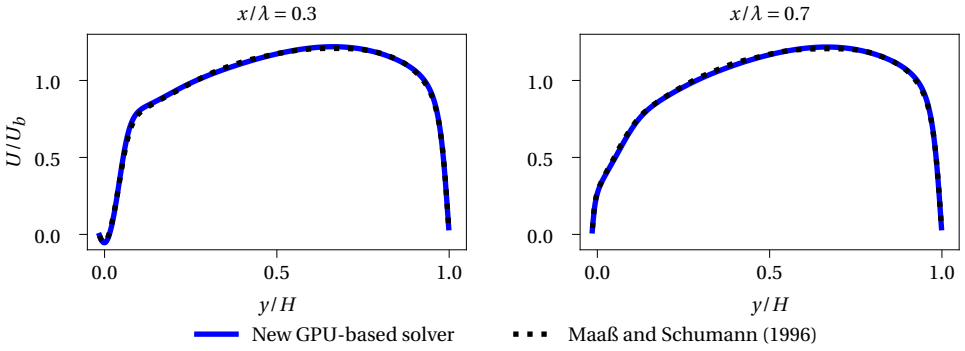


Figure C.2: Validation of the GPU-accelerated solver with respect to the DNS case from (Maaß & Schumann, 1996). Here, H is the height of the channel and λ is the wavelength of the sinusoidal surface.

REFERENCES

Busse, A., Lützner, M., & Sandham, N. D. (2015). Direct numerical simulation of turbulent flow over a rough surface based on a surface scan. *Computers & Fluids*, *116*, 129–147. https://doi.org/https://doi.org/10.1016/j.compfluid.2015.04.008

Maaß, C., & Schumann, U. (1996). Direct numerical simulation of separated turbulent flow over a wavy boundary. In *Flow simulation with high-performance computers ii: Dfg priority research programme results 1993–1995* (pp. 227–241). Vieweg+Teubner Verlag. https://doi.org/10.1007/978-3-322-89849-4%5C_17

Peeters, J., & Sandham, N. (2019). Turbulent heat transfer in channels with irregular roughness. *International Journal of Heat and Mass Transfer*, *138*, 454–467. https://doi.org/https://doi.org/10.1016/j.ijheatmasstransfer.2019.04.013

Thakkar, M., Busse, A., & Sandham, N. (2018). Direct numerical simulation of turbulent channel flow over a surrogate for nikuradse-type roughness. *Journal of Fluid Mechanics*, *837*. https://doi.org/10.1017/jfm.2017.873

LIST OF PUBLICATIONS

JOURNAL PUBLICATIONS

1. **R. Diez**, J.W.R. Peeters and P. Costa, "A pencil-decomposed numerical algorithm for many-GPU calculations of turbulent wall flows at high Reynolds number". Computer Physics Communications, vol. 316, pp. 109811, Nov 2025. (Chapter 2)

2. **R. Diez**, S. Smit, J.W.R. Peeters and R. Pecnik, "Machine learning for RANS turbulence modeling of variable property flows". Computer & Fluids, vol. 255, pp. 105835, Apr 2023. (Chapter 3)

3. G. Otero, A. Patel, **R. Diez** and R. Pecnik, "Turbulence modelling for flows with strong variations in thermo-physical properties". International Journal of Heat and Fluid Flow, vol. 73, pp. 114–123, Oct 2018.

4. **R. Diez**, I. Akkerman and J.W.R. Peeters. "Machine learning for the prediction of the local skin friction factors and Nusselt numbers in turbulent flows past rough surfaces". International Journal of Heat and Fluid Flow, special issue on 12^{th} International Symposium on Turbulence and Shear Flow Phenomena (TSFP12), vol. 103, pp. 109204, Oct 2023. (Chapter 4)

5. **R. Diez** and J.W.R. Peeters, "Data-driven optimization of rough surfaces for convective heat transfer enhancement". International Journal of Heat and Mass Transfer, vol. 251, pp. 127313, Nov 2025. (Chapter 5)

INTERNATIONAL CONFERENCE PRESENTATIONS

1. **R. Diez**, I. Akkerman and J.W.R. Peeters, "Machine learning for the prediction of the local skin friction factors and Nusselt numbers in turbulent flows past rough surfaces", full presentation, in 12^{th} International Symposium on Turbulence and Shear Flow Phenomena (TSFP12), Osaka, Japan (Online), Jul 2022.

2. **R. Diez**, I. Akkerman and J.W.R. Peeters, "Data-driven optimization of rough surfaces for convective heat transfer enhancement", full presentation, in 18^{th} European Turbulence Conference, Valencia, Spain, Sep 2023.

3. **R. Diez**, J.W.R. Peeters and P. Costa, "A pencil-decomposed numerical algorithm for many-GPU calculations of turbulent wall flows at high Reynolds number", full presentation, in 9^{th} European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS), Lisbon, Portugal, Jun 2024.