ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Predicting Turbulent Boundary Layer Flows Using Transformers Coupled to the Multi-Physics Simulation Tool m-AIA

Rakesh Sarma[a,*], Fabian Hübenthal[b], Fabian Orland[c], Christian Terboven[c] and Andreas Lintermann[a]

[a]Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, Wilhelm-Johnen-Straße, 52425 Jülich, Germany
[b]RWTH Aachen University, Institute of Aerodynamics and Chair of Fluid Mechanics, Wüllnerstraße 5a, 52062 Aachen, Germany
[c]RWTH Aachen University, Chair for High-Performance Computing, Seffenter Weg 23, 52074 Aachen, Germany

ARTICLE INFO[†]

Keywords:
Transformers;
Turbulent Boundary Layers;
Computational Fluid Dynamics / Artificial Intelligence Coupling

ABSTRACT

Time-marching of turbulent flow fields is computationally expensive with traditional numerical solvers. In this regard, transformer neural network, which has been largely successful in many other technical and scientific domains, can potentially predict complex flow fields faster compared to physics-based solvers. In this study, a transformer model is trained for a turbulent boundary layer problem, which is then coupled to the multi-physics solver m-AIA to make predictions of velocity fields. The method can potentially contribute to significant reduction in computational effort while maintaining high accuracy.

1. Introduction

Numerical prediction of turbulence remains challenging due to its multiscale nature that requires highly-resolved simulations to accurately capture the temporal and spatial dynamics [1]. In the Computational Fluid Dynamics (CFD) community, numerical solvers capable of simulating complex turbulent dynamics have been developed, albeit demanding substantial computational resources and high resolutions. Alternatively, Machine Learning models, for instance based on Convolutional Neural Networks [2], have emerged as promising alternatives. The use of transformer architecture-based models for time-marching turbulent fields have been limited to few recent successful efforts, often for prediction of compressed representations of the flow field to reduce computational effort [3, 4], which may suppress information of the high-frequency components. Nonetheless, transformers can potentially be an effective neural network to perform complex long-term temporal predictions of turbulent flows, while allowing distributed training given the multi-head attention configuration. In this study, the transformer architecture is applied to the prediction of a Turbulent Boundary Layer (TBL) problem, with a unique reconstruction strategy of the input velocity fields, which allows the prediction of the full velocity field without any intermediate compression step. Furthermore, the trained model is coupled to the highly-parallel multi-physics simulation tool m-AIA formerly known as the Zonal Flow Solver (ZFS) [5], which was further developed towards m-AIA. The coupling framework is needed to couple the physical solver (in this case, m-AIA) to the distributed deep learning inference with the transformer. Replacing m-AIA-based expensive time-marching of the turbulent fields with the transformer model is expected to significantly reduce the computational costs.

2. Turbulent boundary layer problem specification

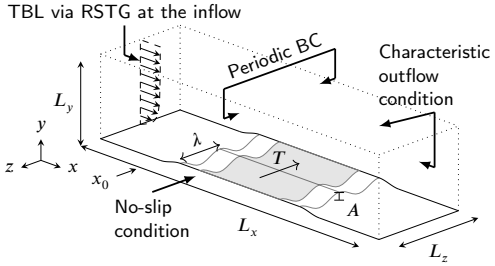


Figure 1: Sketch of the CFD domain with the three-dimensional actuated turbulent boundary layer flow. The gray area indicates the region of interest where the TBL data is extracted.

[†]This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02459 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
r.sarma@fz-juelich.de (R. Sarma); f.huebenthal@aia.rwth-aachen.de (F. Hübenthal); orland@itc.rwth-aachen.de (F. Orland); terboven@itc.rwth-aachen.de (C. Terboven); a.lintermann@fz-juelich.de (A. Lintermann)
ORCID(s): 0000-0002-7069-4082 (R. Sarma); 0009-0000-7159-8220 (F. Hübenthal); 0000-0002-8681-266 (F. Orland); 0000-0003-2284-2957 (C. Terboven); 0000-0003-3321-6599 (A. Lintermann)

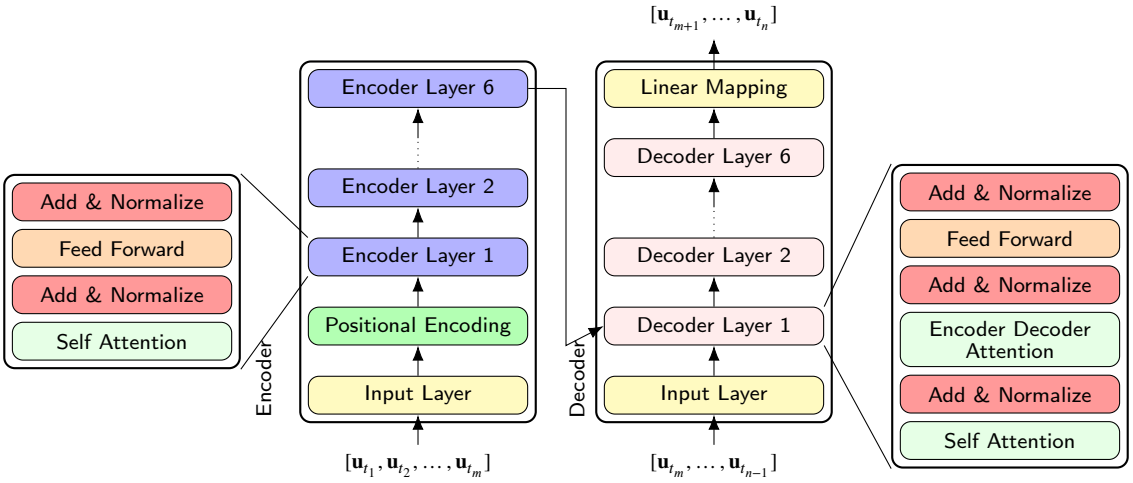


Figure 2: Architecure of the transformer model based on encoder-decoder configuration adapted from [8].

The CFD model is based on a validated Zero-Pressure Gradient flat plate approximation of the active drag reduction technique using spanwise traveling transversal surface waves. Wall-resolved Large-Eddy Simulation (LES) is performed using the in-house CFD solver m-AIA[1] [6, 7]. The physical domain of the flat plate model is given in Fig. 1, where the dimensions in the Cartesian directions are L_x, L_y and L_z. The actuation parameters are the wavelength λ, the time period T and the amplitude A. At the inflow of the domain, the reformulated synthetic turbulence generation (RSTG) method is used to initiate a TBL flow. The onset of the surface actuation, analyzed in [6, 7], is located at x_0, where a fully developed TBL is established. The surface area A_{surf} for the integration of the wall-shear stress τ_w is shaded in gray. Periodic Boundary Conditions (BC) are used in the spanwise direction z, characteristic outflow conditions are applied on the downstream and upper boundaries, and the no-slip condition is imposed on the wall [6].

Further details on the numerical method, the computational setup, validation of the LES and BC can be found in Albers et al. [6].

3. Transformer for temporal prediction

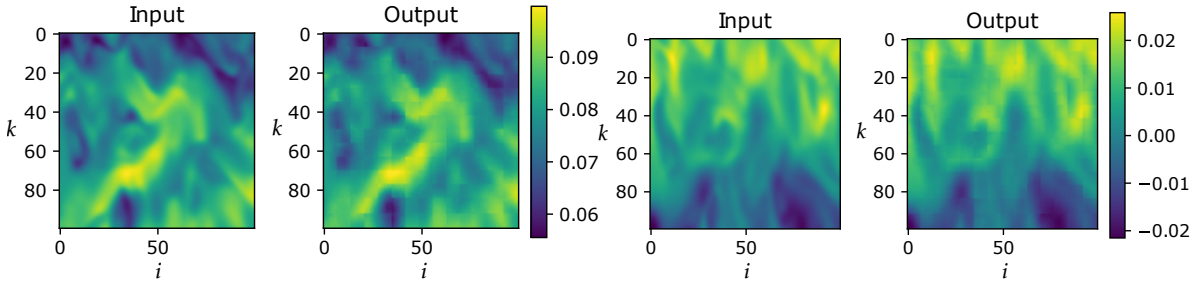
The transformer architecture is adapted from an encoder-decoder configuration, used for temporal predictions [8], which is shown in Fig. 2. The encoder consists of an input layer, positional encoding and a stack of six encoding layers, where each layer consists of a self-attention and a fully connected layer, followed by a normalization layer. The input layer is a fully-connected network and the positional encoding consists of sine and cosine functions. For the decoder layers, there is additional layer to apply self-attention over encoder outputs, where the input is from the encoder output and the decoder output is a linear mapping to the target sequence. Look-ahead masks are applied to ensure that the decoder only sees information from the previous time-steps.

If the training dataset consists of n velocity fields at time-instances $t_1, t_2, \ldots, t_n$, the encoder takes in a sequence of $\mathbf{u}_{t_1}, \mathbf{u}_{t_2}, \ldots, \mathbf{u}_{t_m}$ as input, and the decoder outputs the velocities at time-instances $\mathbf{u}_{t_{m+1}}, \ldots, \mathbf{u}_{t_n}$, where $1 < m < n$. Here, $\mathbf{u}$ is the velocity field vector, m is the encoder sequence length and $n - m$ is the target sequence length. The decoder input in this case would consist of velocities at time-instances $\mathbf{u}_{t_m}, \ldots, \mathbf{u}_{t_{n-1}}$. The input velocity field is reshaped to smaller cubic sub-domains (8^3 for this study), where each sub-domain is treated as a separate batch by the transformer. This allows to limit the number of features that the model needs to predict, thus reducing the complexity of the self-attention mechanism. Furthermore, 16 attention heads are employed and the Adam optimizer is used for training. The trainings are conducted with the DeepSpeed[2] framework in a distributed training setup provided by the AI4HPC[3] library. Exemplary predictions by the transformer model of the streamwise (u) and spanwise (w) components of the velocity field are shown in Fig. 3. It can be seen that the model is able to provide good qualitative predictions of the velocity field. This trained model is then coupled to the m-AIA solver.

[1] m-AIA https://git.rwth-aachen.de/aia/m-AIA/m-AIA

[2] DeepSpeed https://github.com/microsoft/DeepSpeed

[3] AI4HPC https://ai4hpc.readthedocs.io/en/latest/



(a) Streamwise (u) component (b) Spanwise (w) component

Figure 3: Exemplary prediction (Output) from the transformer of TBL velocity field slice in the wall-normal direction on the i-k plane, compared to the original LES field (Input), where i is the streamwise and k is the spanwise direction. In this case, Input and Output refer to the LES-predicted and transformer-predicted velocity field at a future time-step.

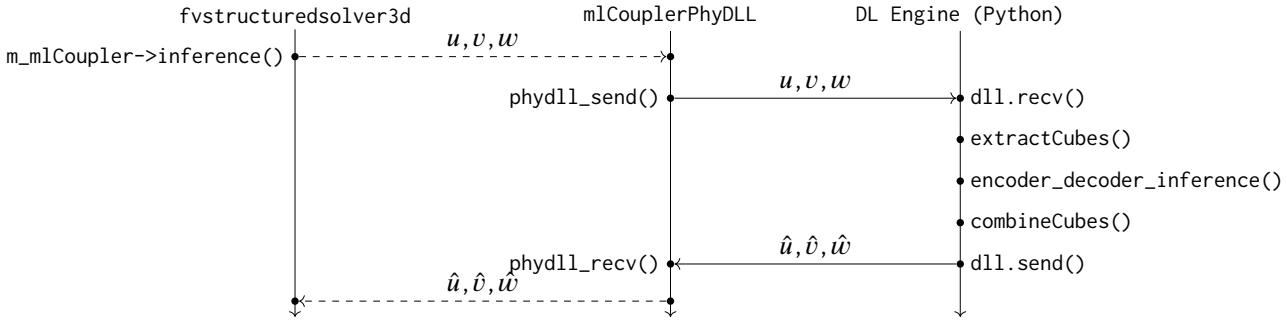


Figure 4: Coupling workflow between m-AIA and PhyDLL.

4. Coupling m-AIA with the transformer

Coupling m-AIA with the trained transformer is achieved using the open-source "Physics Deep Learning coupLer" (PhyDLL)[4], which provides a coupling mechanism for parallel physical solvers via the Message Passing Interface (MPI) using the Multiple Program Multiple Data paradigm. It establishes a mapping between processes running the physical solver and processes via a user-defined DL Engine. In this work, the DL Engine is a Python script, which implements the inference of the transformer. To communicate data between the physical solver and the DL Engine, PhyDLL offers a Fortran, C, and Python interface around MPI's point-to-point operations.

The coupling workflow is illustrated in Figure 4. Inside m-AIA's 3D structured solver (`fvstructuredsolver3d`), **u** is passed to the `mlCouplerPhyDLL` as three one-dimensional fields (u,v,w). This class is derived from an abstract class following the popular strategy design pattern proposed by Gamma et al. [9] to keep the solver independent from the actual coupling algorithm. The `mlCouplerPhyDLL` uses PhyDLL's API to send the flow fields to the DL Engine. First, the 3D flow field is divided into cubic subdomains inside the function `extractCubes()`. The DL Engine manages a ring buffer, storing the data from the m most recent timesteps, to construct sequences of cubic subdomains as input to the transformer. After the input has been constructed, the inference of the transformer is implemented as proposed by Ludvigsen[5]. Multiple forward passes through the transformer are performed to create a suitable target sequence of length $n-m$, which the model can decode to make the final prediction. Afterwards, the function `combineCubes()` combines the cubic subdomains from the final prediction to form the full flow field ($\hat{u}$,$\hat{v}$,$\hat{w}$) again. The final predicted flow field is then sent back to the `mlCouplerPhyDLL` of m-AIA using PhyDLL's API. Finally, the flow field in m-AIA is updated and the simulation continues based on the transformer's prediction.

5. Conclusions

The manuscript explored the use of a transformer model to perform coupled predictions of TBL velocity fields with the CFD solver m-AIA. For this, a transformer model is trained with full velocity field data that is restructured into

[4] PhyDLL https://phydll.readthedocs.io/en/latest

[5] https://github.com/KasperGroesLudvigsen/influenza_transformer

smaller cubic sub-domains. The trained model, which shows good qualitative agreement, is being coupled to the m-AIA solver using the PhyDLL framework, which is an ongoing work at the time of writing this abstract. Therefore, so far qualitative analyses are performed based on the comparison of outputs from the CFD solver and the transformer model. The proposed methodology promises to accurately predict turbulent fields, while significantly reducing the computational effort for time-marching TBL fields by reducing the number of time-steps computed by the physical solver. For a single time step, computational savings of upto 53 times was possible during transformer inference, while also reducing the memory consumption by 1,100 times. As an outlook for the conference talk, quantitative analyses are performed to assess the prediction accuracy also for the coupled configurations and the speed-up compared to the stand-alone CFD solver. Furthermore, the conservation properties (in terms of mass and momentum) of the time-marched velocity fields obtained through the interaction of the CFD solver and the transformer model will be analyzed, and the imbalances will be quantified in future work.

Acknowledgements

The research leading to these results has been conducted in the CoE RAISE project, which receives funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733.

References

[1] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence Modeling in the Age of Data, Annual Review of Fluid Mechanics 51 (2019) 357–377. doi:10.1146/annurev-fluid-010518-040547.

[2] Y. Wang, Z. Xie, K. Xu, Y. Dou, Y. Lei, An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning, Neurocomputing 174 (2016) 988–998. doi:10.1016/j.neucom.2015.10.035.

[3] A. Solera-Rico, C. Sanmiguel Vila, M. Gómez-López, Y. Wang, A. Almashjary, S. T. M. Dawson, R. Vinuesa, β-variational autoencoders and transformers for reduced-order modelling of fluid flows, Nature Communications 15 (2024) 1361. doi:10.1038/s41467-024-45578-4.

[4] M. Z. Yousif, M. Zhang, L. Yu, R. Vinuesa, H. Lim, A transformer-based synthetic-inflow generator for spatially developing turbulent boundary layers, Journal of Fluid Mechanics 957 (2023) A6. doi:10.1017/jfm.2022.1088.

[5] A. Lintermann, M. Meinke, W. Schröder, Zonal Flow Solver (ZFS): a highly efficient multi-physics simulation framework, International Journal of Computational Fluid Dynamics 34 (7-8) (2020) 458–485. doi:10.1080/10618562.2020.1742328.

[6] M. Albers, P. S. Meysonnat, D. Fernex, R. Semaan, B. R. Noack, W. Schröder, Drag reduction and energy saving by spanwise traveling transversal surface waves for flat plate flow, Flow, Turbulence and Combustion 105 (1) (2020) 125–157. doi:10.1007/s10494-020-00110-8.

[7] D. Fernex, R. Semaan, M. Albers, P. S. Meysonnat, W. Schröder, B. R. Noack, Actuation response model from sparse data for wall turbulence drag reduction, Physical Review Fluids 5 (7) (2020) 073901. doi:10.1103/PhysRevFluids.5.073901.

[8] N. Wu, B. Green, X. Ben, S. O'Banion, Deep transformer models for time series forecasting: The influenza prevalence case, ArXiv 2001.08317 (2020). doi:10.48550/arXiv.2001.08317.

[9] E. Gamma, R. Helm, R. Johnson, J. M. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, 1st Edition, Addison-Wesley Professional, 1994.