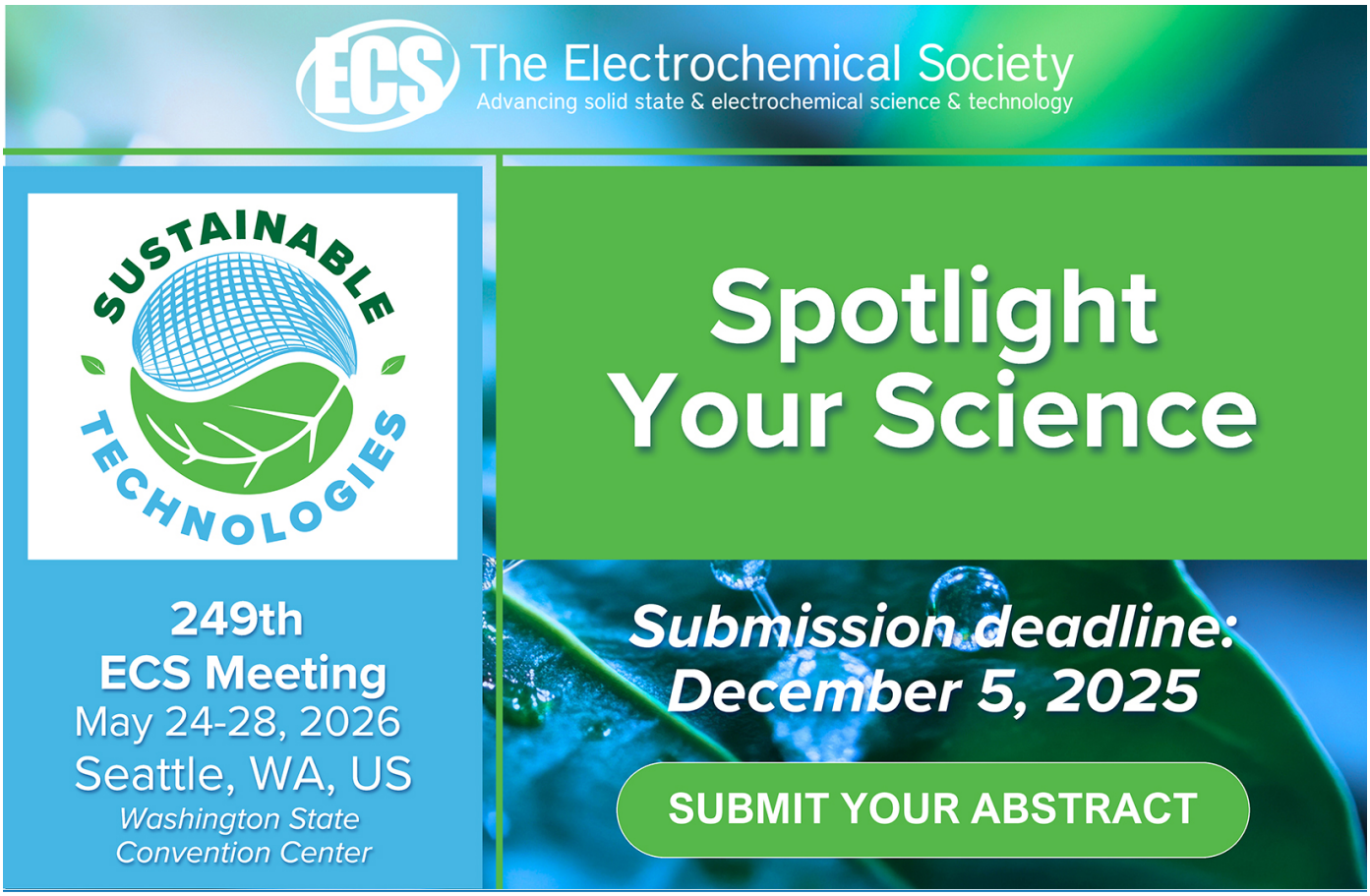
PAPER • **OPEN ACCESS**

Accelerating high order discontinuous Galerkin solvers using neural networks: Wall bounded flows

To cite this article: Oscar A. Mariño *et al* 2024 *J. Phys.: Conf. Ser.* **2753** 012023



Accelerating high order discontinuous Galerkin solvers using neural networks: Wall bounded flows

Oscar A. Mariño[1], David Mayoral[1], Adrián Juanicotena[1], Fernando Manrique de Lara[1], Esteban Ferrer[1,2]

[1] ETSIAE-UPM-School of Aeronautics, Universidad Politécnica de Madrid, Plaza Cardenal Cisneros 3, E-28040 Madrid, Spain
[2] Center for Computational Simulation, Universidad Politécnica de Madrid, Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain

E-mail: esteban.ferrer@upm.es

Abstract. High order solvers are accurate but computationally expensive as they require small time steps to advance the solution in time. In this work we include a corrective forcing to a low order solution to improve the accuracy while advancing in time with larger time steps, and achieve fast computations. The work uses a discontinuous Galerkin framework, where the polynomial order, inside each mesh element, can be varied to provide low or high accuracy. The corrective forcing is included for each high order Gauss nodal point in the mesh. This work is a continuation of [1, 2], where we extend the methodology to wall bounded flows. Namely, we adapt the methodology to a turbulent channel at $Re_\tau = 182$. In this case, we use three neural networks to correct different regions of the flow, which are distinguished by their $y+$ distance to the wall. The methodology is able to correct the low resolution simulation to attain flow statistics that are comparable to high order simulations. We include comparisons for the mean, Reynolds stresses and shear stress on the wall. We achieve good predictions using the corrected low order solution, in mean velocity and its corresponded fluctuations, as well as the shear stress on the wall.

1. Introduction

This paper is a continuation of [1, 2], where we presented neural network accelerations for the simulations of Burger's equations and the Navier-Stokes equations, using the 3D Taylor Green Vortex (TGV) problem. Our proposed methodology evolves a solution with a low-order polynomial while including a corrective forcing to obtain a more accurate solution. The corrective forcing is obtained using deep fully connected neural networks. The methodology proved to be useful, as it was able to correct the solution outside the time frame considered for training. Note that a high order solution was only necessary for training but not after, which allowed accelerations. In [2], we compared the accuracy provided by the methodology with simulations with varying polynomial orders (P) and showed an effective acceleration of 4-5, when correcting a P=3 solution using (only during training) a P=8 simulation. The resulting corrected order of accuracy was P=6.

Neural networks (NN) are gaining popularity in scientific computing as they can complement computational fluid dynamic methodologies; see reviews [3, 4, 5]. A fully connected neural network consists of multiple interconnected layers of artificial neurons, or units, organized into

an input layer, one or more hidden layers, and an output layer. The information flows forward through the network, from the input layer to the output layer, without any loops or connections between neurons within the same layer.

Combining computational fluid dynamics (CFD) with neural networks (NN) has been explored predominantly in the context of low-order methods. Bar-Sinai *et al.* [6] utilized NNs to estimate spatial derivatives of the Burgers' equation on a low-resolution grid. In the field of turbulence modeling, Kochkove *et al.* [7] employed NNs to reconstruct low-fidelity simulations of 2D turbulence, highlighting the potential of CFD and NN integration. Guastoni *et al.* [8, 9] and Güemes *et al.* [10] have also applied NNs, specifically convolutional neural networks (CNNs) and generative adversarial networks (GANs), to spatially interpolate/extrapolate wall quantities in turbulent regimes and generate high-resolution models from low-resolution data. Moreover, researchers have employed NNs to develop turbulent closure models or as substitutes for conventional large eddy simulation (LES) models, as reviewed by Stachenfeld *et al.* [11], Beck and Kurz [12], and Duraisamy [13].

An alternative perspective on the methodology is to regard the low order simulation as a reduced order model, which is augmented through the utilization of a neural network (NN)-based corrective forcing, similar to the approach proposed in [14, 15]. In our research, the reduced order is low order simulation, and consequently retains the efficiency and physic-nature since we solve the Navier-Stokes equations on a coarse grid. This physics-based reduced order model offers flexibility in handling arbitrary boundary conditions, while achieving enhanced accuracy when combined with the corrective forcing provided by the neural network.

In this work, we first summarise the TGV problem, at $Re = 1600$, using more elements than in [2] and without using turbulent models. This problem has a simple set up but provides a complex flow behaviour including and initial laminar regime, followed by transition to turbulence, and finally a fully turbulent regime, as time progresses. In this case the neural network needs to extrapolate the solution in time, to obtain a useful corrective forcing.

Second, we extend the methodology to wall bounded flows. We test the methodology in a turbulent channel at $Re_\tau = 182$. This is a very well known case, that makes use of periodicity in the stream-wise and span-wise direction, but uses solid walls in the normal direction. In this second case, the neural network needs to interpolate (flow is homogeneous in time) to obtain the corrective forcing, but needs to face strong anisotropies in the wall normal direction.

The rest of the paper is organised as follows. First we summarise the methodology to obtain the corrective forcing, including details for wall bounded flows. In section 3, we present results for the TGV problem and the channel, to finalise with conclusions and outlooks.

2. Methodology

Consider a High Order (HO) Discontinuous Galerkin (DG) discretization of the 3D Navier-Stokes (NS) equations. The system is represented as:

$$\frac{d\boldsymbol{q}_{HO}}{dt} = \boldsymbol{p}^{HO}(\boldsymbol{q}_{HO}). \tag{1}$$

Here, $\boldsymbol{q}_{HO} \in \mathbb{R}^{dof^{HO}}$ contains the state variables of the compressible NS, including $(\rho, \rho u, \rho v, \rho w, \rho e)^T$ for all dof^{HO} in the DG mesh. The discrete operator $\boldsymbol{p}^{HO} : \mathbb{R}^{dof^{HO}} \rightarrow \mathbb{R}^{dof^{HO}}$ approximates the NS operator. The degrees of freedom are given by $dof^{HO} = N_{el} \times (P^{HO}+1)^3$, where N_{el} is the number of elements in the mesh and P is the polynomial order.

We introduce a filter operator $\boldsymbol{G} : \mathbb{R}^{dof^{HO}} \rightarrow \mathbb{R}^{dof^{LO}}$ in (1), to obtain:

$$\frac{d\overline{\boldsymbol{q}}_{HO}}{dt} = \boldsymbol{G}\boldsymbol{p}^{HO}(\boldsymbol{q}_{HO}), \tag{2}$$

where $\overline{\boldsymbol{q}}_{HO} \in \mathbb{R}^{dof^{LO}}$ is a low order solution obtained once filtered, and is expected to be more accurate than a low order solution; and $dof^{LO} = N_{el} \times (P^{LO}+1)^3$. We also define a low order evolution equation:

$$\frac{d\boldsymbol{q}_{LO}}{dt} = \boldsymbol{p}^{LO}(\boldsymbol{q}_{LO}). \tag{3}$$

In general, $\boldsymbol{q}_{LO}$ and $\boldsymbol{q}_{HO}$ do not provide the same solution or accuracy, as $\boldsymbol{G}\boldsymbol{p}^{HO}(\boldsymbol{q}_{HO}) \neq \boldsymbol{p}^{LO}(\boldsymbol{q}_{LO})$, due to the non-linearity of the NS operator. To approximate the high order filtered solution $\boldsymbol{q}_{NN} \approx \boldsymbol{q}_{HO}$ using a low order evolution operator, we introduce a corrective forcing term. We rewrite Eq. (3) as:

$$\frac{d\boldsymbol{q}_{NN}}{dt} = \boldsymbol{p}^{LO}(\boldsymbol{q}_{NN}) + \boldsymbol{s}(\boldsymbol{q}_{NN}, \boldsymbol{q}_{HO}). \tag{4}$$

Here, $\boldsymbol{q}_{NN} \in \mathbb{R}^{dof^{LO}}$ represents the new variable obtained by adding the corrective forcing $\boldsymbol{s}(\boldsymbol{q}_{NN}, \boldsymbol{q}_{HO})$. The corrective forcing $\boldsymbol{s}(\boldsymbol{q}_{NN}, \boldsymbol{q}_{HO}) : \mathbb{R}^{dof^{LO}} \times \mathbb{R}^{dof^{HO}} \rightarrow \mathbb{R}^{dof^{LO}}$ depends on both the low and high order solutions and corrects the low order operator to retrieve a high order filtered evolution in time.

The explicit expression for the corrective forcing is given by:

$$\boldsymbol{s}(\boldsymbol{q}_{NN}, \boldsymbol{q}_{HO}) = \boldsymbol{G}\boldsymbol{p}^{HO}(\boldsymbol{q}_{HO}) - \boldsymbol{p}^{LO}(\boldsymbol{q}_{NN}). \tag{5}$$

An alternative expression to Eq. (5) can be derived by expanding the time derivatives in Eq. (3) and Eq. (4): $\boldsymbol{s}(\boldsymbol{q}_{NN}, \boldsymbol{q}_{HO}) = (\overline{\boldsymbol{q}}_{HO}^{n+1} - \boldsymbol{q}_{NN}^{n+1})/\Delta t_n$, and can used to train the neural network. The corrective forcing captures the high order modes and the nonlinear interactions between the low and high order solutions. It is modeled using a deep neural network in this work.

To advance the high and low order corrected systems in time, we consider a temporal discretization. For example, the corrected system Eq. (4) becomes:

$$\boldsymbol{q}_{NN}^{n+1} = \boldsymbol{q}_{NN}^{n} + \Delta t \left[\boldsymbol{p}^{LO}(\boldsymbol{q}_{NN}^{n}) + \boldsymbol{s}^{n}(\boldsymbol{q}_{NN}^{n}, \boldsymbol{q}_{HO}^{n})\right]. \tag{6}$$

Here, $\Delta t = t_{n+1} - t_n$ represents the time step size. The methodology can be applied to other explicit time marching schemes, although a third-order Runge-Kutta scheme is used in this work. This allows us to improve the accuracy of a low order solution by training a neural network with the high order solution and using it to correct the low order solution at future times when the high order solution is unknown.

2.1. High order discontinuous Galerkin solver: HORSES3D

The methodology has been implemented in our high-order spectral element solver HORSES3D [16]. HORSES3D is a 3D parallel framework developed at ETSIAE-UPM (the School of Aeronautics of the Polytechnic University of Madrid). This framework uses the high-order discontinuous Galerkin spectral element method (DGSEM) and is written in modern Fortran 2003. It targets simulations of fluid-flow phenomena such as those governed by compressible Navier-Stokes equations (used in this case) and high order polynomials of arbitrary order in each mesh element.

2.2. Corrective Forcing from Neural Networks

A corrective forcing $\boldsymbol{s}$ is constructed for each degree of freedom in the low order mesh, corresponding to each Gauss point resulting in $(P_{LO}+1)^3$ degrees of freedom per element.

We choose a fully connected Neural Network (NN) for its efficiency and simplicity, although other alternatives such as convolutional or recursive networks could have been considered. The

NN consists of N_{la} layers ($N_{la} = 5$ in the figure), with the last layers using a linear activation function, the first and all intermediate hidden layers using the rectified linear unit (ReLU) activation function, as shown in Figure 1.

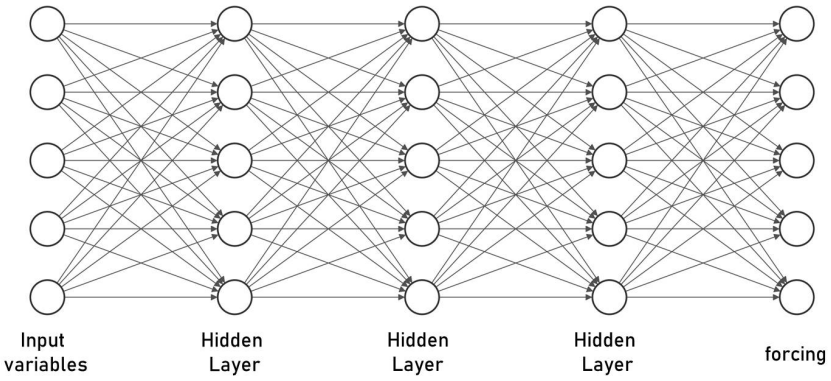


Figure 1: Fully connected deep neural network scheme. The input to the NN is a vector containing the nodal values for ρu, ρv, and ρw inside the element, and the output is a vector containing the nodal values for the forcing (s_2, s_3, and s_4) inside the element.

Regarding the state vectors, we conducted a preliminary test where all variables (density, momentum, and energy) were included in the training (we used the compressible solver HORSES3D). However, we found that a valid corrective forcing could not be obtained. In our test case, which considers incompressible flows with Mach number $M_0 = 0.08$ and 0.2 for the TGV and channel respectively, there exists a large disparity of scales between density/energy and momentum. To address this issue, we only use the three components of momentum for training and re-scale the values at each node. Consequently, the corrective forcing only consists of three components: $\boldsymbol{s} = (0, s_2, s_3, s_4, 0)^T$.

2.3. Isotropic algorithm

We choose to have the same NN architecture for every element and use all elements to train the network. This approach is advantageous in transitional/turbulent cases as in the Taylor Green Vortex case, that does not exhibit large variations between elements (e.g., no walls). Therefore, all elements can contribute to the training, as an element can represent the flow field of another element shifted in time or space.

However, if different flow structures are present in each mesh element, as in the channel case, then using all elements for training may not provide an accurate corrective forcing. In these cases, we divide the channel in layers (separate by $y+$ values) to enhance the training. Thus, while the methodology presented is independent of the flow physics, the NN training needs to consider the underlying flow physics.

The number of time steps used for each phase of the algorithm are summarized in the following table 1.

Regarding the remaining training hyperparameters, our primary emphasis was not on enhancing the overall efficiency of the algorithm, but rather on refining its precision. To achieve this, we employed an extensive number of training epochs, implemented a gradual reduction in learning rates over time, all with the overarching objective of minimizing the loss function to its utmost extent.

Table 1: TGV cases studied

Case	t= 4.5s	t=9s	t= 15s
HO phase			
Start(s)	4.5	9	15
HO iterations	600	600	600
LO & filtered HO iterations	200	200	200
Finish(s)	4.62	9.12	15.12
Training Phase			
Batches size	20	20	20
Epochs	30	30	30
Training time-steps	200	200	200
Training percentage (%)	90	90	90
LO correction phase			
Start(s)	4.5	9	15
LO iterations	1500	1000	1000
Finish(s)	5.4	9.6	15.6

2.4. Wall bounded flow training algorithm

While the Taylor–Green Vortex present time evolving flow structures (laminar to turbulent flow), the channel flow presents flow structures are consistent in time but significant variations can now be observed in space and between mesh elements. Indeed, anisotropy is introduced by the walls in the normal direction.

Previous knowledge about channel flows tells us that turbulence in boundary-layer tends to follow the same pattern and exhibits similar velocity profiles in the normal direction. Experimental evidence and DNS data show that three different regions can typically be identified within the boundary layer, namely: viscous sublayer, buffer layer and a "logarithmic-law region". Taking this into consideration, we divide the channel into several layers in the normal direction to the wall to train each section of the channel with a separate NN, so that we can capture specific physics within the corrective forcing related to the region. This resulted in effective corrective forcings. Note that the strong time-evolution of flow structures characteristic of the Taylor–Green Vortex is not present in the channel, and therefore there is no need for constant re-training of the NNs. Once trained, the corrective forcings are valid for all times.

In the following sections, we use the vertical coordinate to limit the three different layers for the whole height of the channel. We set the limits at $y^+_{limtit_1} \sim 8.6$ and, $y^+_{limtit_2} \sim 36$ in the first half of the channel. We replicate the same limits at $y^{top}_{limtit_i} = 2\delta - y^{bottom}_{limtit_i}$, such that the same wall distance in maintained. Thus, the channel is divided in 6 layers, 3 for each half. As the NN have no information on the spatial position, we combine the elements of the layers symmetrically to train each network, such that we have 3 NN, the first one for the elements near both walls, the second for the elements at mid distance, and the last one for the centre of the channel.

Due to the strong anisotropy observed in the channel case, the MLP architecture being used for the Taylor–Green Vortex problem has not proven to be able to capture the complex flow phenomena near the walls, even after assigning one dedicated NN to each boundary-layer region. The solution proposed here is to increase the number layers in each NN. The NNs with 20 layers provide reliable results while, in terms of computational cost, they do not involve much more training time than the previous 4-layer NNs.

3. Results

3.1. The Taylor Green Vortex Problem

Numerical experiments were conducted on the Taylor–Green Vortex (TGV) [17] for Reynolds numbers: $Re = 1600$. The TGV problem has been extensively used to study numerical methods

and analyze their ability to reproduce various flow regimes[18]. The TGV configuration consists of a three-dimensional periodic box $[-\pi, \pi]^3$ with the following initial condition:

$$\begin{aligned} \rho &= \rho_0, \\ v_1 &= V_0 \sin x \cos y \cos z, \\ v_2 &= -V_0 \cos x \sin y \cos z, \\ v_3 &= 0, \\ p &= \frac{\rho_0 V_0^2}{\gamma M_0^2} + \frac{\rho_0 V_0^2}{16}(\cos 2x + \cos 2y)(\cos 2z + 2), \end{aligned} \tag{7}$$

where ρ_0, V_0, γ, and M_0 are constants representing density, velocity, specific heat ratio, and Mach number, respectively.

All simulations were performed using a coarse h-mesh with 16^3 elements and polynomial orders ranging from $P = 2$ to 8, as illustrated in Figure 2. Further details about our specific DG formulations can be found in [16]. We employed the Roe scheme for advective fluxes, the Bassi-Rebay 1 scheme for viscous fluxes, and a third-order three-stage Runge-Kutta scheme for time integration. The flow is considered incompressible with a Mach number of $M_0 = 0.08$ in all simulations. The reported times are non-dimensional and have been scaled with the characteristic velocity V_0 and the reference length $L = 1$.

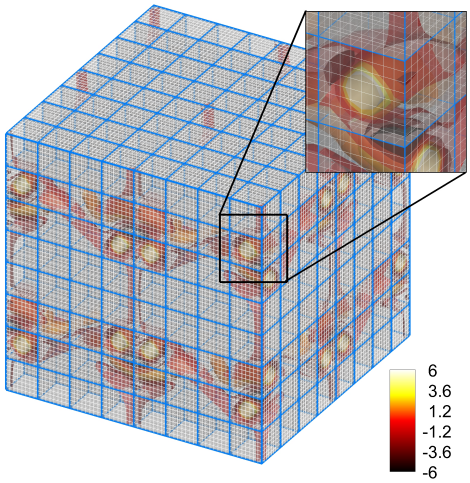


Figure 2: TGV problem: Mesh and Q-criterion iso-surfaces for Re=1600 at $t = 7$.

To validate the methodology, we show the time evolution of the L_2 norm for the difference between the high order solution $P_{HO} = 8$ and low order solutions (with and without corrective forcing) $P_{LO} = 2$. To compute this norm, we filter the high order solution and compare it with the low order solution as time progresses.

We check our methodology for three non-dimensional times. The first one corresponds to laminar flow structure ($t^* = 4.5$), the second one to a transitional regime $t^* = 9$ and the third one to a fully turbulent regime $t^* = 15$. For all times we test two NN architectures by increasing the number of layer from 4 to 20. In all cases, we observe that the NN-corrected simulation provides lower errors after training, but there is a clear dependency on the length of the correction depending on flow regime. In addition, using 20 layer clearly improves the predictions. We have also tested 30 layers but did not see a clear advantage (results not shown). It is noteworthy that the case where $t = 4.5s$ stands out as the most optimal scenario for our architecture. Conversely,

when we scrutinized the outcomes across other time intervals, particularly in the instance where $t = 15s$, we discerned a marked decline in accuracy. We have observed improvements when using more complex neural network architectures (CNN or LSTM), and are currently preparing a detailed publication on the topic.

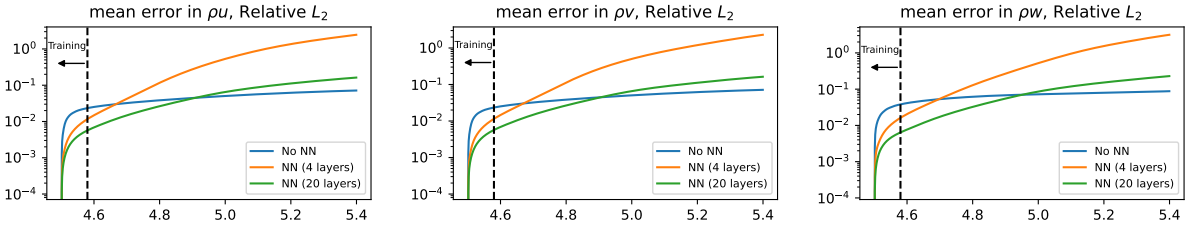


Figure 3: $\mathcal{L}_2$ error curves for different MLP architectures for $t^* = 4.5$.

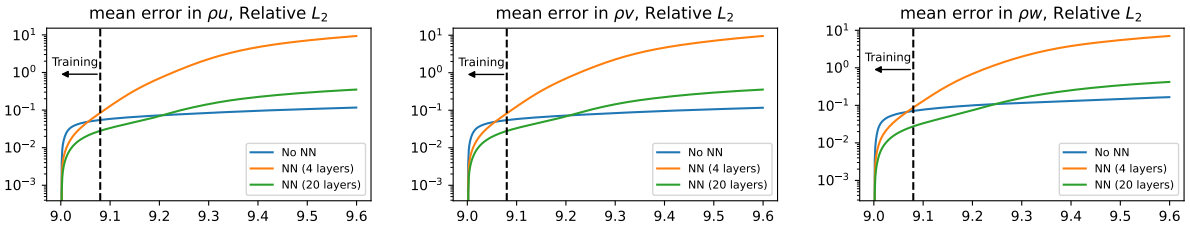


Figure 4: $\mathcal{L}_2$ error curves for different MLP architectures for $t^* = 9$.

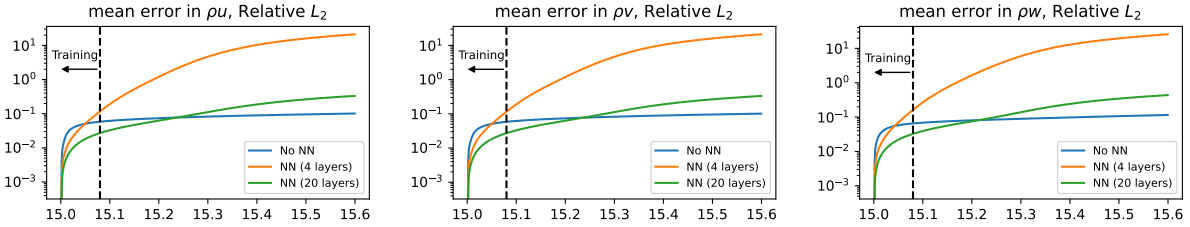


Figure 5: $\mathcal{L}_2$ error curves for different MLP architectures for $t^* = 15$.

3.2. Channel

The second case is a turbulent channel flow at friction Reynolds number $Re_\tau = 182$ and Mach number $M_0 = 0.2$. The friction Reynolds number is defined as $Re_\tau = u_\tau \delta / \nu$, where u_τ is the characteristic friction velocity, δ is channel half height and ν is the kinematic viscosity. The channel geometry is set as $6\delta \times 2\delta \times 3\delta$ in the x, y, z axis respectively. We employ adiabatic no-slip wall boundary conditions at the top and bottom of the domain ($y = 0$ and $y = 2\delta$), and periodic boundary conditions in the stream-wise and span-wise direction. To force the motion of the fluid in the stream-wise direction, we apply a constant force in this direction, equal to the theoretical shear stress at the wall. The settings for the numerical discretization are the same as the TGV case, namely, Roe and BR1 for the convective and viscous fluxes respectively and RK3 for the time integration.

We use a wall-resolving grid (first node from the wall is located at $y^+ = 1$) to avoid turbulence models or wall functions. The h-mesh consists of 1078 elements distributed as $11 \times 14 \times 7$ in the domain. The elements are evenly distributed in the x and z direction, while having stretching in the y direction, such that there are more elements in the wall regions. The fine simulation uses a polynomial order $P = 4$, while the low solution $P = 2$. We set a constant time step throughout the entire simulation, at a value of $\Delta t = 2 \times 10^{-3} \delta/U$ for the low order solution, while for the high order we use $\Delta t = 5 \times 10^{-4} \delta/U$, such that there are exactly 4 high order time steps per one low order step.

An initial stage of 30 flow-through units ($T = L_x/U$) is simulated to ensure a fully developed turbulent flow. Then, we train for 1000 low order time steps each NN corresponding to one of the three layers of the channel, as explained in Section 2.4. Finally, we run it during a time span of $33T$, where the turbulence statistics are recorded. A snapshot of the stream-wise velocity is presented in Figure 6, along with the h-mesh. Turbulence fluctuations can be observed in the figure.

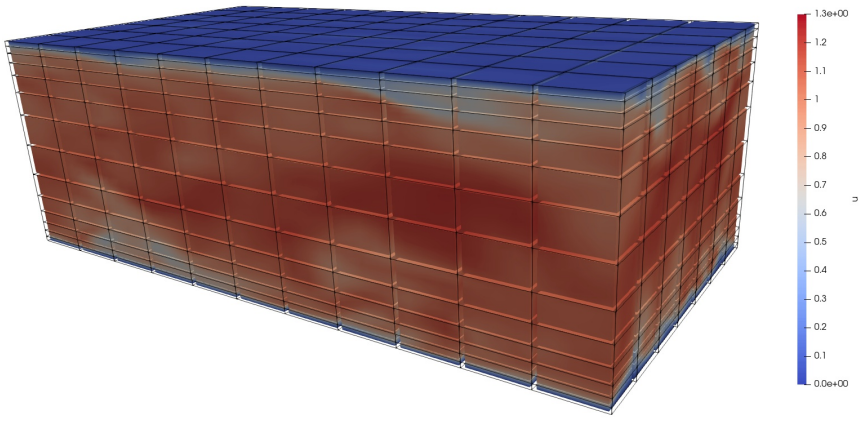


Figure 6: Channel flow: Instantaneous snapshot of stream-wise velocity for the high order $P = 4$ solution.

The mean stream-wise velocity profile in wall units is shown in Figure 7, for the high order and low order solution, as well as the corrected low order using 4 and 20 neuron layers. The high order solution is in very good agreement with the very well resolved DNS results of [19], while the low order solution deviates for $y^+ > 10$. In addition, the solution corrected with 4 neuron layers shows little difference with respect to the uncorrected low order solution. The solution corrected with 20 neuron layers is able to predict with accuracy the mean velocity, obtaining small disagreement after $y^+ = 30$, obtaining a close match to the high order $P = 4$ solution.

Similar behavior is found for the Reynolds stresses, depicted in Figure 8. Good agreement between the reference DNS simulation and the high order solution is apparent. When considering the variance of stream-wise velocity fluctuations (left figure), the low order solution is very different from the DNS solution while the NN corrected solution provides reasonable agreement. When considering the covariance of stream and wall-normal fluctuations of velocity (right figure), the low order corrected solution approaches the high order solution, improving significantly the low order results. Further improvements can be achieved by using more elaborated architectures or by increasing the number of layers/neurons, but the presented results show clearly the high potential of the proposed methodology also for wall bounded flows.

Another variable of interest is the wall shear stress (τ_w), related with the friction velocity (u_τ) and thus with the actual numerical friction Reynolds number (Re_τ). In table 2 the values obtained by each simulation are summarised and compared with the theoretical one. Very good

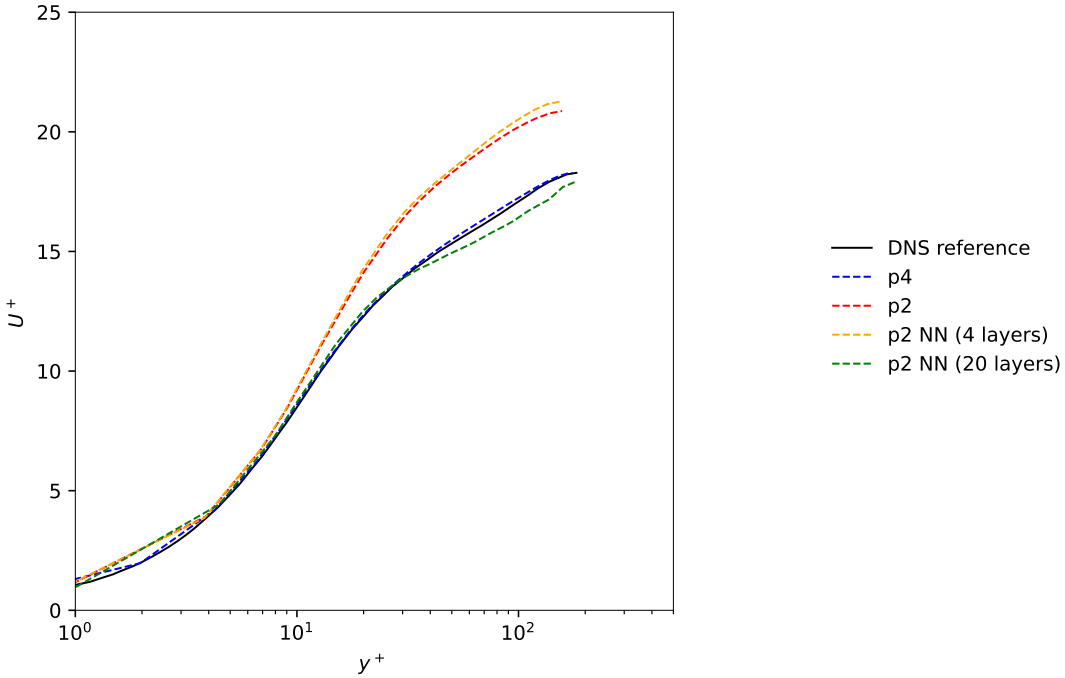


Figure 7: Mean stream-wise velocity profile for high order, low order and corrected low order compared with reference DNS results [19]

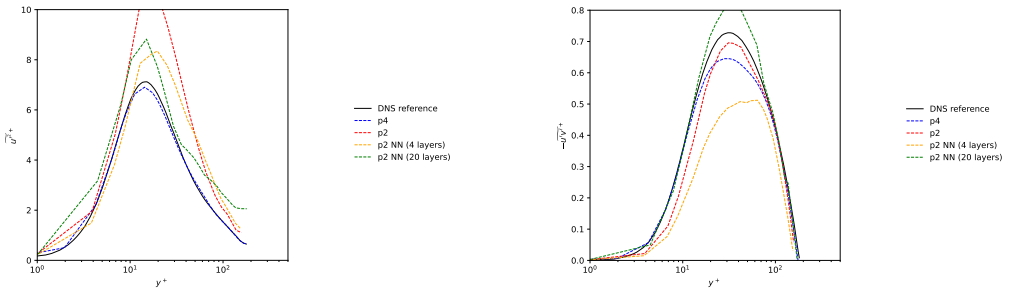


Figure 8: Variance of stream-wise velocity (left) and covariance of stream-wise and wall-normal velocity (right) for high order, low order and corrected low order compared with reference DNS results [19]

agreement is found for the high order solution and the one corrected with 20 neuron layers, with less than 3% error in both cases, while for the low order solution and the corrected using 4 layers, the friction Reynolds number is under-predicted, showing an error of 11%.

Table 2: Theoretical and calculated Reynolds numbers

Theoretical	HO (p4)	LO (p2), uncorrected	LO (p2), 4-layer NN (p2)	LO (p2), 20-layer NN
182	178	163	160	186

To assess the performance of the three neural networks, the s_2 component of the corrective forcing, associated to stream-wise momentum has been plotted along the wall-normal direction. The high order filtered solution has been plotted as a reference, since this is the aim to be

achieved by the corrective forcing. Dashed lines divide the regions allocated to each neural network.

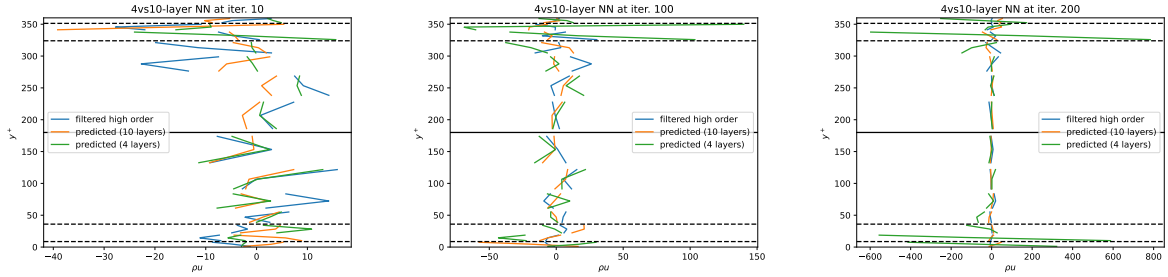


Figure 9: Resolved, filtered ρu and predicted ρu by the 4- and 10-layer NNs along the wall-normal direction, at iterations 10, 100 and 200. Dashed lines divide the regions allocated to each neural network.

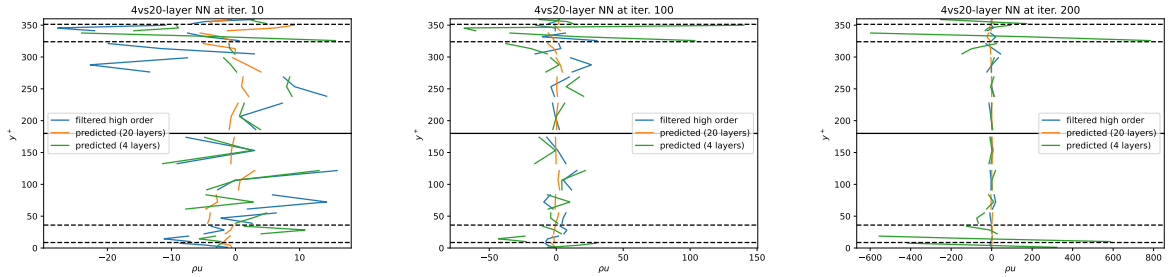


Figure 10: Resolved, filtered ρu and predicted ρu by the 4- and 20-layer NNs along the wall-normal direction, at iterations 10, 100 and 200. Dashed lines divide the regions allocated to each neural network.

While 4-layer NNs were already delivering satisfactory predictions in the center of the channel, results near walls were far from accurate and were being responsible for the divergence of the simulation after training. When using larger layers in the neural network, a substantial improvement near walls is achieved, although the performance at the channel center is not necessarily enhanced. Even though 10-layer neural networks seem to provide promising results, the simulations were diverging after long times. Non-divergence of the simulations is only ensured when using 20 neuron layers. Performance with larger neural networks is to be further investigated.

For completeness, Figure 11 shows the training and validation losses of the different neural networks used. A global loss decrease both during the training and validation proves a correct training process.

4. Conclusions

The application of neural networks (NNs), has shown great potential in accelerating high-order solvers for turbulent flows. NNs provide a flexible framework for learning complex relationships between inputs and outputs, making them suitable for constructing corrective forcings in high-order simulations. By training NNs on high-fidelity data, they can effectively capture the underlying flow physics and provide accurate corrections to low-order simulations.

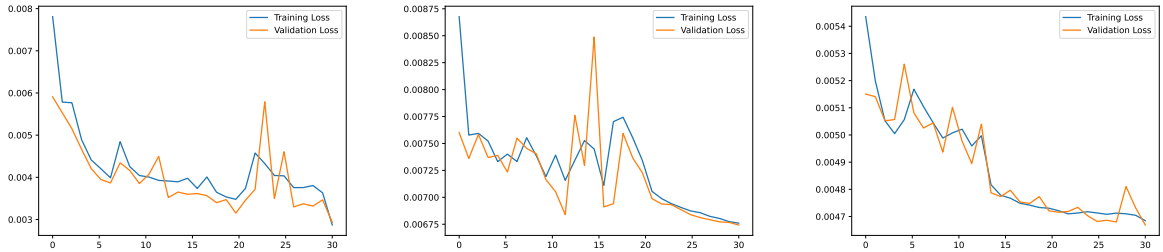


Figure 11: Training and validation losses of each neural network.

The use of NNs, offers several advantages in accelerating high-order solvers for turbulent flows. They can significantly reduce computational costs while enhancing accuracy. Furthermore, our proposed methodology can handle complex flow features, such as transitional flows, turbulence and now wall bounded flows.

Using the mathematical and physical knowledge of the flow provides a good basis to generate a good algorithm for the training of the NN. On the one hand, for temporal evolving flows it is necessary to retrain the network prove to be a good approach. On the other hand, for anisotropic features in space it proves useful to create a dedicated network for different regions.

It is important to note that the successful implementation of NN, in turbulent flow simulations relies on appropriate data generation, network architecture design, and training strategies. Proper validation and verification procedures are necessary to ensure the reliability and accuracy of the results. Continued research and development in this field are expected to further accelerate high order methods for more complex flows.

Acknowledgments

OM and EF would like to thank the support of Agencia Estatal de Investigación (for the grant "Europa Excelencia 2022" Proyecto EUR2022-134041/AEI/10.13039/501100011033) and the Mecanismo de Recuperación y Resiliencia de la Unión Europea. EF knowledges the help of the Comunidad de Madrid and Universidad Politécnica de Madrid for the Young Investigators award: APOYO-JOVENES-21-53NYUB-19-RRX1A0. This research has received funding from the European Commission through the H2020-MSCA-ITN-209 project zEPHYR (grant agreement No 860101) and the European Union (ERC, Off-coustics, project number 101086075). Views and opinions expressed here are those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. This work was performer in part during the Fifth Madrid Summer Workshop, funded by the European Research Council under the Caust grant ERCAdG-101018287. Finally, all authors gratefully acknowledge the Universidad Politécnica de Madrid (www.upm.es) for providing computing resources on Magerit Supercomputer.

References

[1] Manrique de Lara F and Ferrer E 2022 *Computers & Fluids* **235** 105274
[2] Manrique de Lara F and Ferrer E 2023 *J. Comput. Phys.* **489** 112253
[3] Brunton S L and Kutz J N 2019 *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press)
[4] Vinuesa R and Brunton S L 2022 *Nature Computational Science* **2** 358–366
[5] Le Clainche S, Ferrer E, Gibson S, Cross E, Parente A and Vinuesa R 2023 *Aerosp. Sci. Technol.* **138** 108354
[6] Bar-Sinai Y, Hoyer S, Hickey J and Brenner M P 2019 *P. Nat. Acad. Sci. USA* **116** 15344–15349

[7] Kochkov D, Smith J A, Alieva A, Wang Q, Brenner M P and Hoyer S 2021 *P. Nat. Acad. Sci. USA* **118** e2101784118
[8] Guastoni L, Güemes A, Ianiro A, Discetti S, Schlatter P, Azizpour H and Vinuesa R 2021 *J. Fluid Mech.* **928** A27
[9] Balasubramanian A G, Guastoni L, Schlatter P, Azizpour H and Vinuesa R 2023 (*Preprint* ArXiv:2303.00706)
[10] Güemes A, Discetti S, Ianiro A, Sirmacek B, Azizpour H and Vinuesa R 2021 *Phys. Fluids* **33** 075121
[11] Stachenfeld K, Fielding D B, Kochkov D, Cranmer M, Pfaff T, Godwin J, Cui C, Ho S, Battaglia P and Sanchez-Gonzalez A 2021 Learned coarse models for efficient turbulence simulation (*Preprint* ArXiv:2112.15275)
[12] Beck A and Kurz M 2021 *GAMM-Mitteilungen* **44** e202100002
[13] Duraisamy K 2021 *Phys. Rev. Fluids* **6**(5) 050504
[14] Fabra A, Baiges J and Codina R 2022 *Computer Methods in Applied Mechanics and Engineering* **399** 115280
[15] Dar Z, Baiges J and Codina R 2023 *Computer Methods in Applied Mechanics and Engineering* **415** 116232
[16] Ferrer E, Rubio G, Ntoukas G, Laskowski W, Mariño O, Colombo S, Mateo-Gabín A, Marbona H, Manrique de Lara F, Huergo D, Manzanero J, Rueda-Ramírez A, Kopriva D and Valero E 2023 *Computer Physics Communications* **287** 108700
[17] Taylor G I and Green A E 1937 *Proceedings of the Royal Society of London Series A* **158** 499–521
[18] Brachet M E, Meiron D I, Orszag S A, Nickel B G, Morf R H and Frisch U 1983 *J. Fluid Mech.* **130** 411–452
[19] Lee M and Moser R D 2015 *J. Fluid Mech.* **774** 395–415