Machine learning for RANS turbulence modeling of variable property flows

Sanhueza, Rafael Diez; Smit, Stephan H.H.J.; Peeters, Jurriaan W.R.; Pecnik, Rene

DOI
10.1016/j.compfluid.2023.105835

Publication date
2023

Document Version
Final published version

Published in
Computers and Fluids

Citation (APA)
Sanhueza, R. D., Smit, S. H. H. J., Peeters, J. W. R., & Pecnik, R. (2023). Machine learning for RANS turbulence modeling of variable property flows. *Computers and Fluids*, *255*, Article 105835. https://doi.org/10.1016/j.compfluid.2023.105835

Computers and Fluids 255 (2023) 105835

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Machine learning for RANS turbulence modeling of variable property flows

Rafael Diez Sanhueza *, Stephan H.H.J. Smit, Jurriaan W.R. Peeters, Rene Pecnik

Process and Energy Department, Delft University of Technology, Leeghwaterstraat 39, 2628 CB Delft, The Netherlands

ARTICLE INFO

Keywords:
Turbulence modeling
Machine learning
Variable properties

ABSTRACT

This paper 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.

1. Introduction

1.1. Turbulence modeling

The governing equations of fluid flow have long been established, yet modeling 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 [1,2], such as supercritical fluids or hypersonic flows. Understanding the behavior 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 [3–7]. 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 [8]. This result is known as Morkovin's hypothesis [9]. DNS of compressible channel flows [10] 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. [11] has provided a mathematical basis for the use of the semi-local scaling as proposed by Huang et al. [12]. 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

$$Re_\tau^\star \equiv \frac{\sqrt{\overline{\rho}/\overline{\rho}_w}}{\overline{\mu}/\overline{\mu}_w} Re_\tau, \tag{1}$$

* Corresponding author.
E-mail address: R.G.DiezSanhueza-1@tudelft.nl (R.D. Sanhueza).

https://doi.org/10.1016/j.compfluid.2023.105835
Received 29 April 2022; Received in revised form 29 November 2022; Accepted 15 February 2023
Available online 17 February 2023

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. [13], 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 and Larsson [14]. Second, this insight has later been used in Pecnik and Patel [3] 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. [1] 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.

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. [15]. While different ML methods are available, deep neural networks have emerged as one of the most promising alternatives to improve turbulence modeling [16]. 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 [17]. 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. [18], who developed deep neural networks to model turbulence with embedded Galilean invariance, or in the work of Parish and Duraisamy [19], 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 and Duraisamy [19]. 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 and Duraisamy [19] and the subsequent publications of Singh et al. [20,21,22]. 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.

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.

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 [3,13,23]. 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 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. [3,13,23]. The second set of DNS consists of high-Mach number compressible channel flow simulations with air modeled as a calorically perfect gas [14] (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 1 as well. The third set of simulations contains incompressible channel flows [24] (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 [11]. 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 [10,11].

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

Table 1
DNS database of turbulent channel flows with variable properties (low-Mach) [3, 13], with ideal gases at high-Mach numbers [14], and with constant properties (incompressible) [24].

Number	Case ID	ρ	μ	λ	$Re_{\tau,w}$	Pr_w	$Ec_{\tau,w}$	ϕ
Low-Mach number cases [3,13,23]								
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 [14]								
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 [24]								
26	$IC.Re180$				180			
27	$IC.Re550$				550			
28	$IC.Re950$	–	–	–	950	–	–	–
29	$IC.Re2000$				2000			
30	$IC.Re4200$				4200			

Reynolds/Favre averaged streamwise momentum and energy equations for a fully developed turbulent channel flow read

$$\frac{\partial}{\partial y}\left[\left(\frac{\mu}{Re_{\tau,w}}+\mu_t\right)\frac{\partial u}{\partial y}\right]=-1, \tag{2}$$

$$\frac{\partial}{\partial y}\left[\left(\frac{\lambda}{Re_{\tau,w}Pr_w}+\frac{c_p\mu_t}{Pr_t}\right)\frac{\partial T}{\partial y}\right]=-Ec_{\tau,w}\left(\frac{\mu}{Re_{\tau,w}}+\mu_t\right)\left(\frac{\partial u}{\partial y}\right)^2-\frac{\phi}{Re_{\tau,w}Pr_w}, \tag{3}$$

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

$$Re_{\tau,w}=\frac{\rho_w u_{\tau,w} h}{\mu_w},\quad Pr_w=\frac{\mu_w c_{p,w}}{\lambda_w},\quad Ec_{\tau,w}=u^2_{\tau,w}/(c_{p,w}T_w)=(\gamma-1)\,Ma^2_{\tau,w}, \tag{4}$$

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. (2) and (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) [25], which has also been used in our previous studies to model turbulence in variable property turbulent channel flows [1]. The equations for the turbulent kinetic energy k and turbulent dissipation ε read

$$\underbrace{\mu_t\left(\frac{\partial u}{\partial y}\right)^2}_{P_k}-\underbrace{\rho\,\varepsilon}_{D_k}+\underbrace{\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial y}\right]}_{T_k}=0, \tag{5}$$

$$\underbrace{C_{\varepsilon 1}\,P_k\,\frac{\varepsilon}{k}}_{P_\varepsilon}-\underbrace{C_{\varepsilon 2}\,f_\varepsilon\,\rho\,\frac{\varepsilon^2}{k}}_{D_\varepsilon}+\underbrace{\frac{\partial}{\partial y}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial y}\right]}_{T_\varepsilon}=0, \tag{6}$$

with the supporting damping functions

$$f_\varepsilon=\left(1-\frac{2}{9}e^{-(Re_t/6)^2}\right)\left(1-e^{-y^\star/5}\right)^2,\qquad f_\mu=\left(1-e^{-y^\star/70}\right)\left(1+\frac{3.45}{\sqrt{Re_t}}\right), \tag{7}$$

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

$$Re_t=\frac{\rho\,k^2}{\mu\,\varepsilon},\quad y^\star=y^+\sqrt{\frac{\rho}{\rho_w}}\frac{\mu_w}{\mu},\quad \mu_t=C_\mu\,f_\mu\,\rho\,\frac{k^2}{\varepsilon}. \tag{8}$$

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 [1]. 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 [26] 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 [27].

The velocity profiles for a few selected cases with the original MK turbulence model are shown in Fig. 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 $JFM.CRe^{\star}_{\tau}$ from Table 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. 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 and Duraisamy [19], which is also suitable to account for turbulence in variable-property flows.

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. (5) and (6), can then be written as

$$P_k-\beta_k D_k+T_k=0, \tag{9}$$

$$P_\varepsilon-\beta_\varepsilon D_\varepsilon+T_\varepsilon=0. \tag{10}$$

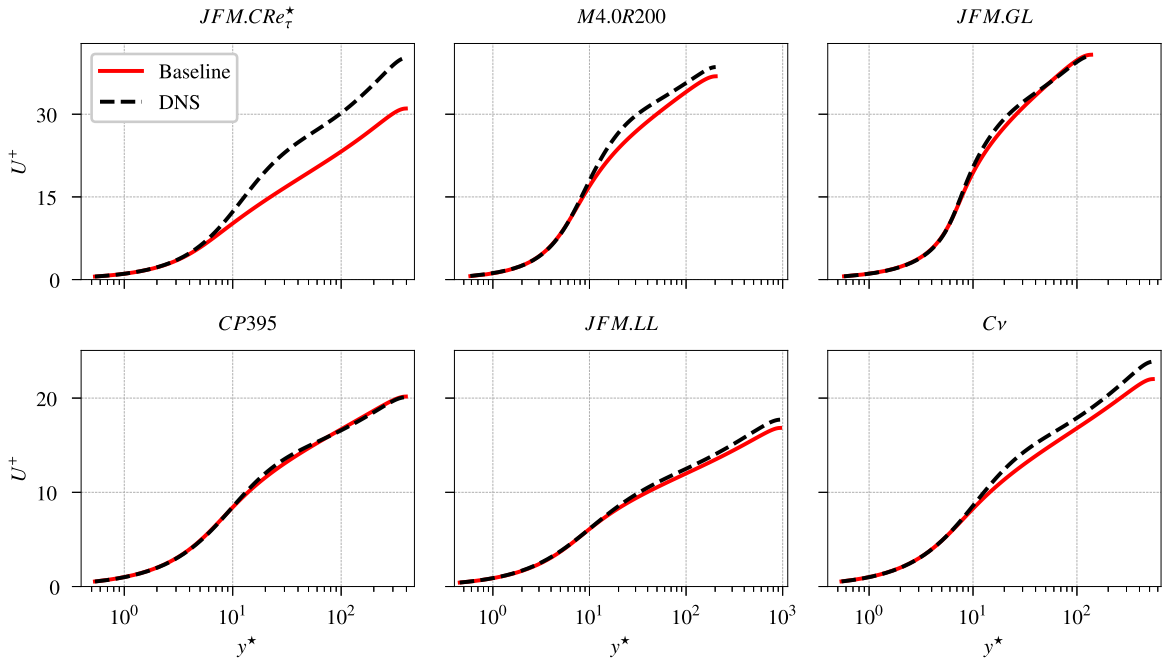

Fig. 1. Velocity profiles obtained with the original MK turbulence model using the density and viscosity profiles obtained from DNS. The black dashed lines correspond to the DNS data, whereas the red solid lines correspond to the RANS simulations.

Contrary to Parish and Duraisamy [19], we multiply the dissipation rather than the production terms, in order to adhere to energy conservation, i.e. turbulent kinetic production also appears in the mean kinetic energy equation with an opposite sign [28]. 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 $\mathcal{J}$ is defined as

$$\mathcal{J} = \sum_{i=1}^{N} I_U \left(\frac{u_i - u_i^*}{S_U} \right)^2 + I_k \left(\frac{\delta_k}{S_k} \right)^2 + I_\varepsilon \left(\frac{\delta_\varepsilon}{S_\varepsilon} \right)^2, \tag{11}$$

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

$$\delta_k = D_k \left(\beta_k - 1 \right), \tag{12}$$

$$\delta_\varepsilon = D_\varepsilon \left(\beta_\varepsilon - 1 \right). \tag{13}$$

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

$$S_U = max(|u^*|), \tag{14}$$

$$S_k = max \left(|P_k|, \ |D_k|, \ |T_k| \right), \tag{15}$$

$$S_\varepsilon = max \left(|P_\varepsilon|, \ |D_\varepsilon|, \ |T_\varepsilon| \right). \tag{16}$$

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 [29], since it was found that each field inversion shows clear inflection points (discussed in detail later).

3.1.1. 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 $\mathcal{J}$ 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 $\mathcal{J}$.

The GD algorithm used in the present study is based on the traditional bold drive method [30]. However, we also introduced gradient inertia to increase the convergence speed. The final approach for the optimizer is shown in algorithm 1. 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 $\mathcal{J}(\beta')$ 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 ($\mathcal{J}(\beta') > \mathcal{J}_{n-1}$), 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 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.

Algorithm 1 Modified bold drive method with added momentum to accelerate optimization.

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

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 [31,32]. 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.

3.1.2. 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 $\mathcal{R}(W(\beta),\beta) = 0$, that contains one entry per every discretized cell and scalar equation. The variable $W(\beta)$ corresponds to the vector of discretized 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 $\nabla_\beta \mathcal{J}$ can be calculated as

$$\nabla_\beta \mathcal{J} = \Psi^T \cdot \frac{\partial \mathcal{R}}{\partial \beta} + \frac{\partial \mathcal{J}}{\partial \beta}, \tag{17}$$

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

$$\left[\frac{\partial \mathcal{R}}{\partial W}\right]^T \cdot \Psi = -\left[\frac{\partial \mathcal{J}}{\partial W}\right]^T. \tag{18}$$

The main advantage of the discrete adjoint method is that only the vector Ψ must be calculated from Eq. (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 $\partial\mathcal{R}/\partial W$ and $\partial\mathcal{R}/\partial\beta$, we utilize symbolic algebra packages, such as Sympy [33]. 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.

3.2. 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 [34,35]. An early reference to the use of hyperbolic tangent neurons in the context of fluid mechanics can be found in the work of Milano and Koumoutsakos [36]. In the first layer of our neural networks we introduce logarithmic neurons [37], 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 modeling 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:

$$\mathcal{J}_{train} = \frac{1}{N}\underbrace{\sum_{i=1}^{N}\left(\delta_{NN,i} - \delta_{FI,i}\right)^2}_{\|\delta_{error}\|^2} + \lambda\ \frac{1}{M}\underbrace{\sum_{j=1}^{M} w_j^2}_{\|w\|^2}. \tag{19}$$

In Eq. (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 3.4.

3.2.1. 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 modeling 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 [38]. 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

Table 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										
$CP150$				✓					✓	
$CP395$					✓				✓	✓
$CRe^{\star}_{\tau}$										
$SRe^{\star}_{\tau GL}$								✓		
GL							✓			
$LL1$					✓					
$SRe^{\star}_{\tau LL}$		✓						✓		
$SRe^{\star}_{\tau Cv}$.										✓
Cv	✓				✓	✓				
$LL2$				✓						
$CRe^{\star}_{\tau}CPr^{\star}$	✓					✓	✓			✓
$GLCPr^{\star}$		✓					✓		✓	
$V\lambda SPr^{\star}_{LL}$				✓				✓		
$JFM.CRe^{\star}_{\tau}$	✓	✓	✓	✓			✓			✓
$JFM.GL$			✓			✓		✓		
$JFM.LL$			✓		✓				✓	
High-Mach number cases										
$M0.7R400$					✓	✓				✓
$M0.7R600$										
$M1.7R200$		✓					✓			
$M1.7R400$								✓		
$M1.7R600$									✓	
$M3.0R200$	✓			✓						
$M3.0R400$										
$M3.0R600$			✓							
$M4.0R200$	✓	✓	✓			✓				
Incompressible cases										
$IC.Re180$									✓	
$IC.Re550$			✓		✓		✓			
$IC.Re950$		✓		✓						✓
$IC.Re2000$						✓				
$IC.Re4200$								✓		

corresponds to one of the best alternatives available to assess the performance of ML systems trained with small datasets [39].

The test sets for the different K-fold combinations are listed in Table 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 $JFM.CRe^{\star}_{\tau}$ 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 and Hoyas [24] 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 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 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.

3.2.2. 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 behavior, a novel weighted relaxation factor method is introduced, which is able to filter spurious oscillations in the predicted δ corrections. To explain this, we show in Fig. 2 the turbulent kinetic energy budgets (P_k, D_k, T_k) and three profiles of δ corrections for the DNS case $JFM.CRe^{\star}_{\tau}$. 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 $\Delta = 0.6\ y^3\ sin(8\pi y)$. 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

$$\left\|\delta_{ML}\right\| = \alpha\ \left\|\delta_{ini}\right\|, \tag{20}$$

or alternatively,

$$\mathcal{J}_{\delta} = \sum_{i=1}^{N} \delta^2_{ML,i} = \sum_{i=1}^{N} \left(\alpha\ \delta_{ini,i}\right)^2\ (= constant). \tag{21}$$

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. (21) as β times the production term P, namely

$$\delta_{ML} = \beta P \tag{22}$$

in the corresponding turbulence modeling equations. The key idea to build a robust methodology is to recognize that spurious machine learning corrections, such as δ_{ini} in Fig. 2, create large oscillations in the β multipliers defined by Eq. (22). As a result, the introduction of a L2-regularization hyper-parameter, λ, for these β multipliers would immediately penalize the presence of large oscillations in regions where

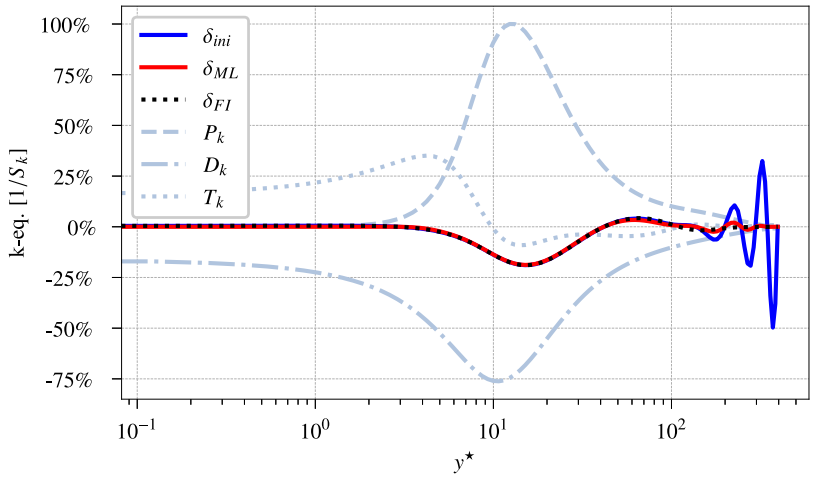

Fig. 2. Turbulence budgets for the optimized k-equation of the MK turbulence model after applying the independent set of δ_k corrections obtained during the field inversion study for the DNS case $JFM.CRe_\tau^\star$. The initial machine learning predictions (δ_{ini}), shown in blue, contain the fictitious perturbation term: $\Delta = 0.6\ y^3\ sin(8\pi y)$. The red line δ_{ML} corresponds to the corrections obtained after applying the weighted relaxation factor methodology.

RANS turbulence models are inactive. The cost function associated with this problem is

$$\begin{aligned} \mathcal{J}_\beta &= \sum_{i=1}^{N} \left(\delta_{ML,i} - \delta_{ini,i}\right)^2 + \lambda\beta_i^2 \\ &= \sum_{i=1}^{N} \left(P_i\beta_i - \delta_{ini,i}\right)^2 + \lambda\beta_i^2. \end{aligned} \tag{23}$$

Eq. (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 $\left\|\beta^2\right\|$ according to a regularization hyper-parameter λ. In order to minimize the cost function defined in Eq. (23), its Jacobian can be forced to form a null vector:

$$\nabla_\beta \mathcal{J}_\beta = 0. \tag{24}$$

Replacing Eq. (23) into the previous condition, yields the following element-wise array equation:

$$P\left(P\beta - \delta_{ini}\right) + \lambda\beta = 0. \tag{25}$$

Re-arranging the terms of Eq. (25) further reveals that

$$\beta = \frac{\delta_{ini}\ P}{\lambda + P^2}. \tag{26}$$

Note, Eq. (26) is evaluated element-wise. Replacing Eq. (26) back into Eq. (22) gives a direct residual equation for λ

$$\mathcal{R}_\lambda = \sum_{i=1}^{N} \left(\delta_{ini,i}\ \frac{P_i^2}{\lambda + P_i^2}\right)^2 - \sum_{i=1}^{N} \left(\alpha\ \delta_{ini,i}\right)^2 = 0. \tag{27}$$

Since Eq. (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 ($\mathcal{R}_\lambda$) is given by the following formula:

$$\nabla_\lambda \mathcal{R}_\lambda = -2 \sum_{i=1}^{N} \frac{\left(\delta_{ini,i} P_i^2\right)^2}{\left(\lambda + P_i^2\right)^3}. \tag{28}$$

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

$$\delta_{ML} = \frac{\delta_{ini}\ P^2}{\lambda + P^2}. \tag{29}$$

Eqs. (27)–(29) constitute the only required components to implement our weighted relaxation factor methodology in a computer environment. The results depicted in Fig. 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. Final framework

The final machine learning framework for the study can be found in Fig. 3, which is split into two stages. In the first stage, shown in Fig. 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(a) and the original approach proposed by Parish & Duraisamy [19] 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(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(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 3.2.2 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(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.

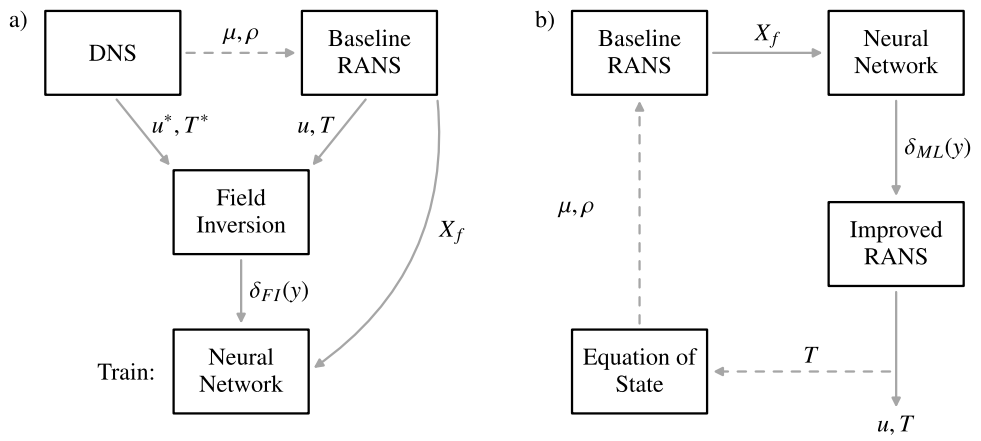

Fig. 3. Final framework established for the FIML methodology. The dashed lines indicate the additional steps which are necessary to handle the presence of variable-property flows, with respect to the original scheme proposed by Parish and Duraisamy [19]. The diagram on the left (a) presents the methodology 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.

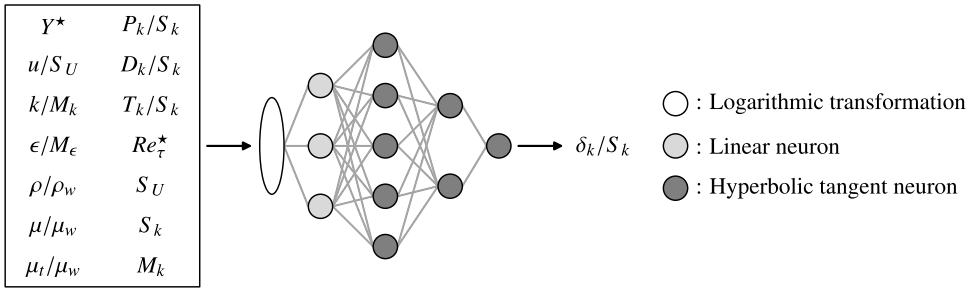

Fig. 4. Neural network architecture created to predict the field inversion corrections (δ_k) required by the MK turbulence model.

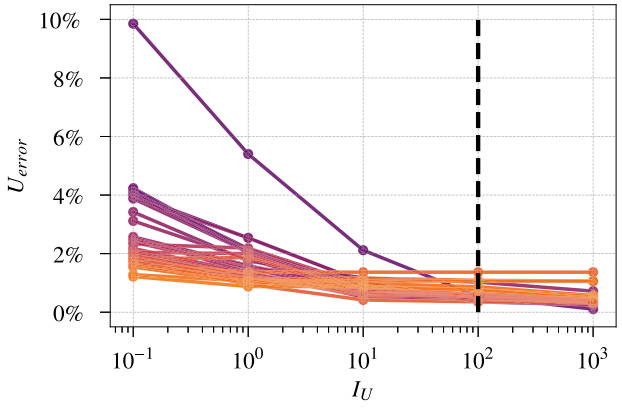

Fig. 5. Application of the elbow method to determine the magnitude of I_U during the initial field inversion study for the MK turbulence model ($I_k = I_\varepsilon = 1$). The black dashed line represents the position where $I_U = 100$.

The final neural network architecture is depicted in Fig. 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. 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:

$$M_\varepsilon = \frac{S_k}{\rho_w}, \quad M_k = \frac{\rho_w M_\varepsilon^2}{S_\varepsilon}. \tag{30}$$

3.4. Interpretation of machine learning results

Regarding the interpretation of the final machine learning results, the integrated gradients (IG) [40] method was used to estimate the importance of every feature passed to the neural network shown in Fig. 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 [40,41]:

$$IG_i = (X_i - X_{i,0}) \int_{\alpha=0}^{1} \frac{\partial}{\partial X_i} \left[\delta(X_0 + \alpha(X - X_0))\right] d\alpha. \tag{31}$$

In Eq. (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.

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. (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 the system. The results obtained can be found in Fig. 5, where it can be

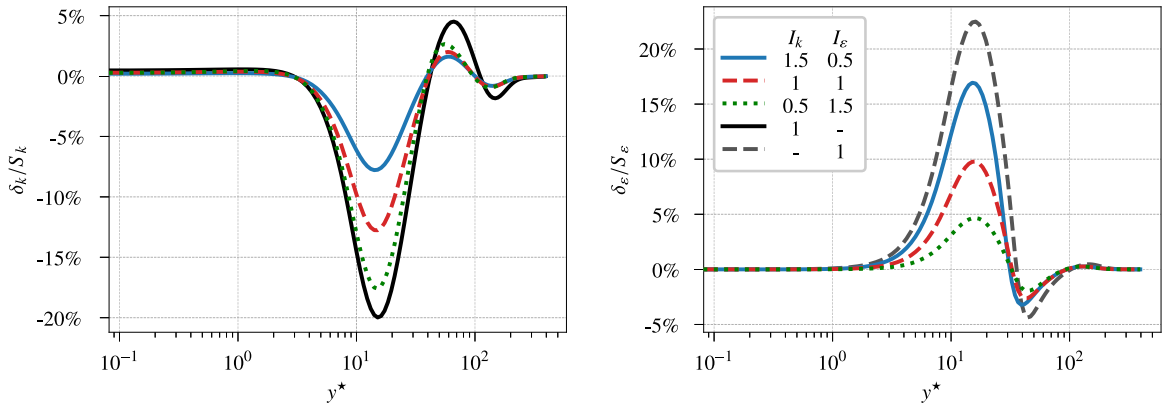

Fig. 6. Effect of the cross-interactions between I_k and I_ε for $I_U = 100$ during the field inversion study for the MK turbulence model considering the DNS case $JFM.CRe_\tau^\star$. The corrections $(I_k, I_\varepsilon) = (1, -)$ and $(I_k, I_\varepsilon) = (-, 1)$ refer to studies where independent sets of δ_k and δ_ε corrections were generated without their counterpart in the MK turbulence model.

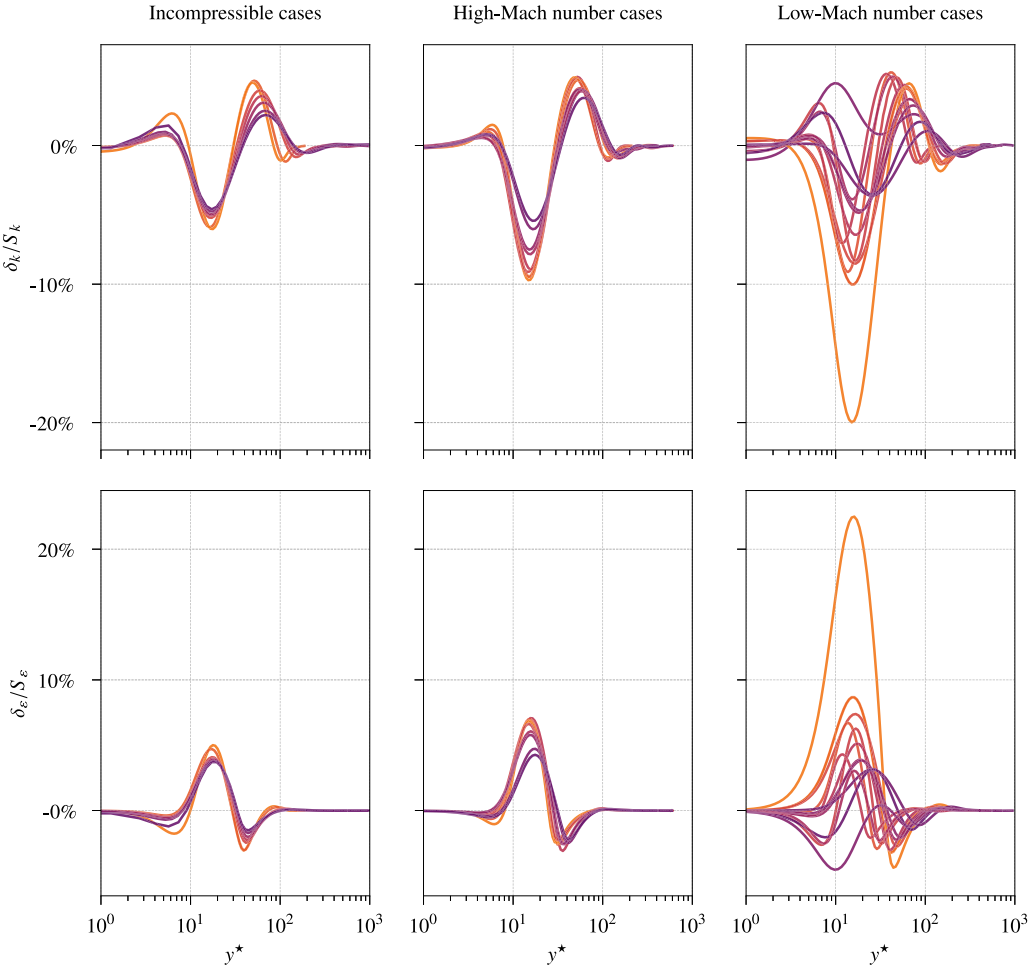

Fig. 7. Independent set of field inversion corrections δ_k and δ_ε obtained for the MK turbulence model while employing $I_U = 100$.

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

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

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. 6 for the DNS case $CRe^{\star}_{\tau}$, which corresponds to the case with the highest modeling 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 results. It was verified that the trends observed in Fig. 6 are also present across all the other DNS cases.

Based on the results presented in Fig. 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 (δ_k, δ_ε) 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. 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. 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.

5. Machine learning predictions

The final ML predictions were obtained by training the neural network architecture described in Fig. 4, and subsequently applying a weighted relaxation factor of $\alpha = 0.95$ in Eq. (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 Table 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 $JFM.CRe^{\star}_{\tau}$ 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 $JFM.CRe^{\star}_{\tau}$ contains with the highest modeling errors with respect to the baseline MK model, and it requires the highest level of δ_k/S_k corrections according to Fig. 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. 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. 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.

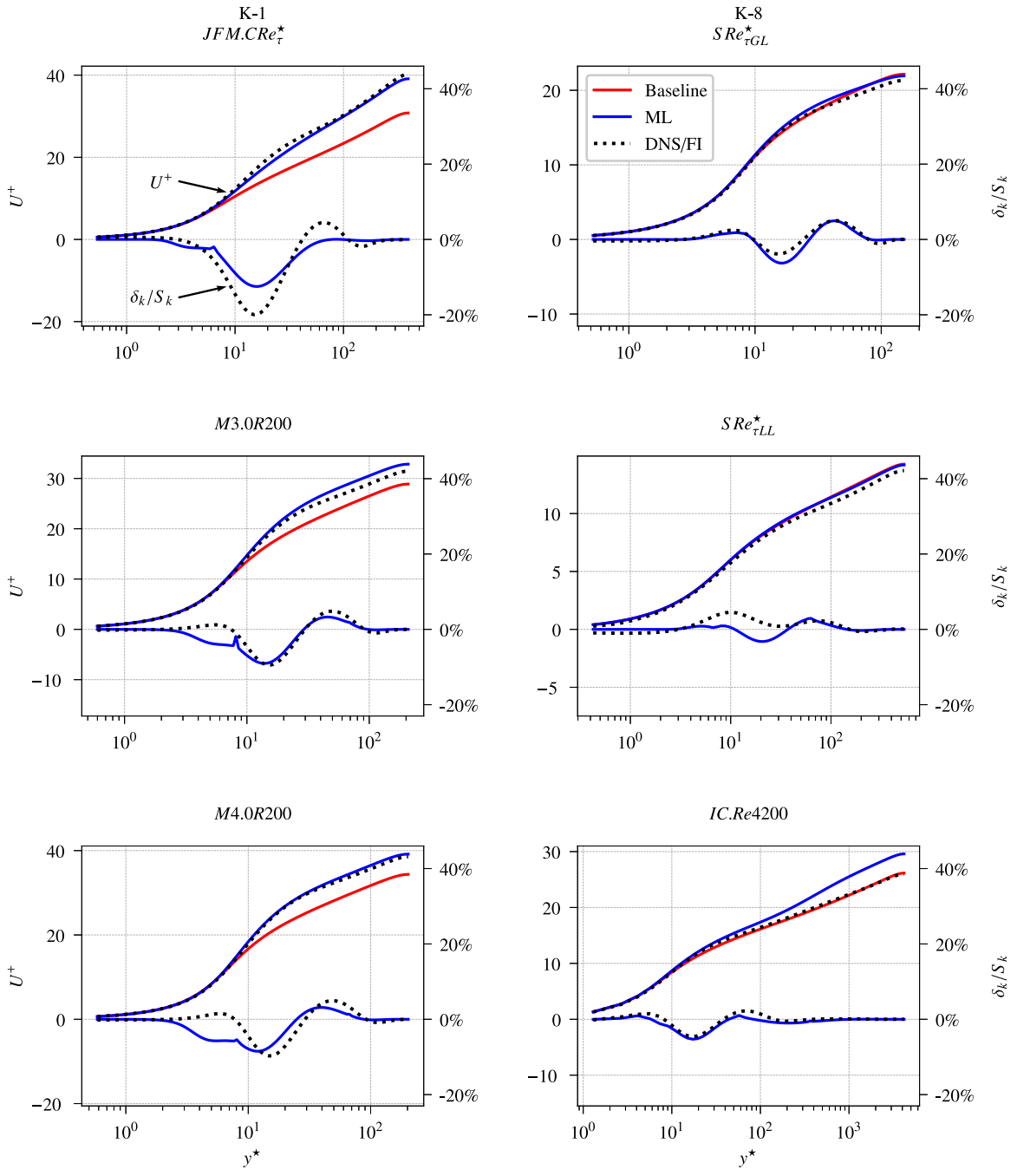

Fig. 8. Selection of results at the extreme ends of the test error ranking for the K-fold validation trials after generating ML predictions for the independent sets of δ_k corrections required by the MK turbulence model. The upper U^+ curves represent the initial RANS velocity profiles (red lines), the 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).

The results for the DNS case $JFM.CRe^{\star}_{\tau}$ are analyzed in greater detail in Fig. 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 $JFM.CRe^{\star}_{\tau}$ 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 $JFM.CRe^{\star}_{\tau}$ 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 modeling conditions.

The results of the non-dimensional feature importance ranking can be found in Fig. 10, which is determined using the integrated gradients (IG) method described in Section 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 features, such as the turbulent production rate P_k/S_k, the turbulent

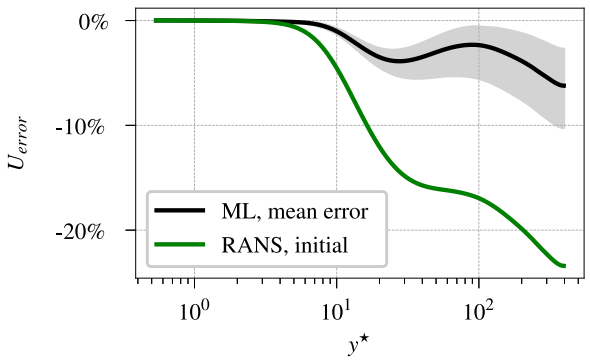

Fig. 9. Results of the uncertainty quantification process followed for the DNS case $JFM.CRe_\tau^\star$ while employing the results of the K-fold validation runs K-1, K-2, K-3, K-4, K-7, K-10 after the prediction of the independent set of δ_k corrections for the MK turbulence model. The gray area corresponds to the maximum deviations observed in the neural network predictions across all K-fold trials.

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. [1].

6. Conclusions

In this paper we used machine learning to improve the predictions of RANS turbulence modeling in channel flows subject to strong variations in their thermophysical properties. The methodology is based on a technique known as FIML proposed by Parish and Duraisamy [19]. 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 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 modeling 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 $y^\star$.

CRediT authorship contribution statement

Rafael Diez Sanhueza: Conceptualization, Methodology, Software, Validation, Investigation, Writing – review & editing. **Stephan H.H.J. Smit:** Methodology, Software, Validation. **Jurriaan W.R. Peeters:** Conceptualization, Methodology, Writing – review & editing. **Rene Pecnik:** Conceptualization, Methodology, Software, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

Rene Pecnik acknowledges the support of the European Research Council through the grant: ERC-2019-CoG-864660, Critical.

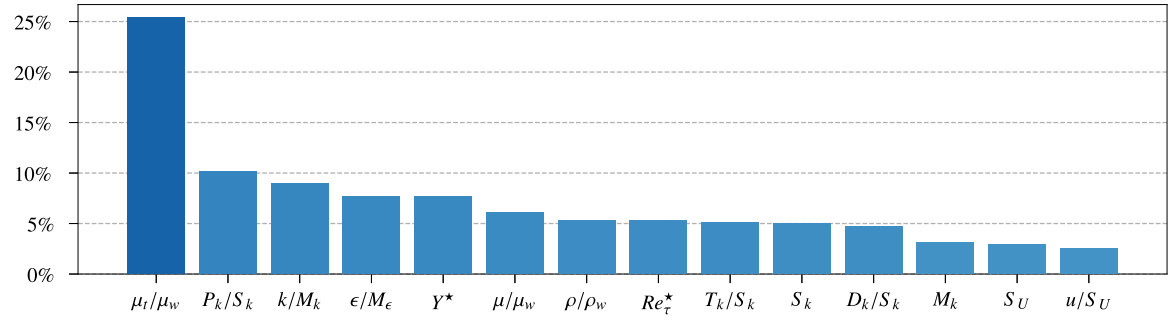

Fig. 10. Feature importance ranking according to the integrated gradients method [40].

References

[1] Rodriguez GJO, Patel A, S. RD, Pecnik R. Turbulence modelling for flows with strong variations in thermo-physical properties. Int J Heat Fluid Flow 2018;73:114–23.

[2] He S, Kim W, Bae J. Assessment of performance of turbulence models in predicting supercritical pressure heat transfer in a vertical tube. Int J Heat Mass Transfer 2008;51(19):4659–75.

[3] Pecnik R, Patel A. Scaling and modelling of turbulence in variable property channel flows. J Fluid Mech 2017;823:R1.

[4] Yoo JY. The turbulent flows of supercritical fluids with heat transfer. Annu Rev Fluid Mech 2013;45(1):495–525.

[5] Peeters J. On the effect of pseudo-condensation on the design and performance of supercritical CO2 gas chillers. Int J Heat Mass Transfer 2022;186:122441.

[6] Nemati H, Patel A, Boersma BJ, Pecnik R. Mean statistics of a heated turbulent pipe flow at supercritical pressure. Int J Heat Mass Transfer 2015;83:741–52.

[7] Peeters JWR, Pecnik R, Rohde M, van der Hagen THJJ, Boersma BJ. Turbulence attenuation in simultaneously heated and cooled annular flows at supercritical pressure. J Fluid Mech 2016;799:505–40.

[8] Smits AJ, Dussauge J-P. Turbulent shear layers in supersonic flow. Springer Science & Business Media; 2006.

[9] Morkovin M. Effects of compressibility on turbulent flows. CNRS; 1961, p. 367–80,

[10] Coleman GN, Kim J, Moser RD. A numerical study of turbulent supersonic isothermal-wall channel flow. J Fluid Mech 1995;305:159–83.

[11] Patel A, Peeters J, Boersma B, Pecnik R. Semi-local scaling and turbulence modulation in variable property turbulent channel flows. Phys Fluids 2015;27(9):095101.

[12] Huang PG, Coleman GN, Bradshaw P. Compressible turbulent channel flows: DNS results and modelling. J Fluid Mech 1995;305:185–218.

[13] Patel A, Boersma BJ, Pecnik R. The influence of near-wall density and viscosity gradients on turbulence in channel flows. J Fluid Mech 2016;809:793–820.

[14] Trettel A, Larsson J. Mean velocity scaling for compressible wall turbulence with heat transfer. Phys Fluids 2016;28(2):026102.

[15] Chang W, Chu X, Binte Shaik Fareed AF, Pandey S, Luo J, Weigand B, Laurien E. Heat transfer prediction of supercritical water with artificial neural networks. Appl Therm Eng 2018;131:815–24.

[16] Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. Annu Rev Fluid Mech 2020;52(1):477–508.

[17] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44.

[18] Ling J, Kurzawski A, Templeton J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J Fluid Mech 2016;807:155–66.

[19] Parish E, Duraisamy K. A paradigm for data-driven predictive modeling using field inversion and machine learning. J Comput Phys 2016;305:758–74.

[20] Singh AP, Duraisamy K, Zhang ZJ. Augmentation of turbulence models using field inversion and machine learning. AIAA scitech forum, American Institute of Aeronautics and Astronautics; 2017.

[21] Singh AP, Medida S, Duraisamy K. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. AIAA J 2017;55(7):2215–27.

[22] Singh AP, Matai R, Mishra A, Duraisamy K, Durbin PA. Data-driven augmentation of turbulence models for adverse pressure gradient flows. AIAA aviation forum, American Institute of Aeronautics and Astronautics; 2017.

[23] Patel A, Boersma BJ, Pecnik R. Scalar statistics in variable property turbulent channel flows. Phys Rev Fluids 2017;2:084604.

[24] Jiménez J, Hoyas S. Turbulent fluctuations above the buffer layer of wall-bounded flows. J Fluid Mech 2008;611:215–36.

[25] Myong H, Kasagi N. A new approach to the improvement of k-ϵ; turbulence model for wall-bounded shear flows. JSME Int J. Ser 2 Fluids Eng Heat Transf Power Combust Thermophys Prop 1990;33(1):63–72.

[26] Patel A. Universal characterization of wall turbulence for fluids with strong property variations (Ph.D. thesis), Delft University of Technology; 2017.

[27] Pecnik R, Rodriguez GJO, Patel A, S. RD. RANS channel. 2018, https://github.com/Fluid-Dynamics-Of-Energy-Systems-Team/RANS_Channel.

[28] Durbin PA, Reif BP. Statistical theory and modeling for turbulent flows. John Wiley & Sons; 2011.

[29] Thorndike RL. Who belongs in the family? Psychometrika 1953;18(4):267–76.

[30] Battiti R. Accelerated backpropagation learning: Two optimization methods. Complex Syst 1989;3(4).

[31] Mitliagkas I, Zhang C, Hadjis S, Ré C. Asynchrony begets momentum, with an application to deep learning. 2016, arXiv e-prints arXiv:1605.09774.

[32] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, CoRR, abs/1412.6980 arXiv:1412.6980 URL http://arxiv.org/abs/1412.6980.

[33] Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, Kumar A, Ivanov S, Moore JK, Singh S, et al. SymPy: symbolic computing in Python. PeerJ Comput Sci 2017;3:e103.

[34] Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signals Systems 1989;2(4):303–14.

[35] Hornik K. Approximation capabilities of multilayer feedforward networks. Neural Netw 1991;4(2):251–7.

[36] Milano M, Koumoutsakos P. Neural network modeling for near wall turbulent flow. J Comput Phys 2002;182(1):1–26.

[37] Hines J. A logarithmic neural network architecture for unbounded non-linear function approximation. In: Neural networks, 1996., IEEE international conference on, Vol. 2. 1996, p. 1245–50.

[38] Mosteller F, Tukey JW. Data analysis, including statistics. In: Lindzey G, Aronson E, editors. Handbook of social psychology, Vol. 2. Addison-Wesley; 1968.

[39] Nie Y, De Santis L, Carratù M, O'Nils M, Sommella P, Lundgren J. Deep melanoma classification with K-fold cross-validation for process optimization. In: 2020 IEEE international symposium on medical measurements and applications (MeMeA). 2020, p. 1–6.

[40] Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks. In: Proceedings of the 34th international conference on machine learning - Volume 70. ICML 17, JMLR.org; 2017, p. 3319–28.

[41] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015, https://www.tensorflow.org/, Software available from tensorflow.org.