Proceedings of the 1st international Symposium on AI and Fluid Mechanics
Paper No S5 P2

Differentiable Learning for Turbulence Modeling: A Gradient-based framework in RANS Simulations

L Saverio[1,2], M A Bucci[1], C Content[2] and D Sipp[2]

[1]Digital Sciences & Technologies Department, Safran Tech, 78114 Chateaufort, France
[2]DAAA, ONERA, F-92190 Meudon, France

Keywords: Turbulence Modeling, Machine Learning, RANS, Automatic Differentiation, Pytorch.

Abstract. This work presents an end-to-end optimization strategy integrating the Reynolds-Averaged Navier-Stokes equations within a differentiable Machine Learning framework. Unlike Field Inversion Machine Learning [4], which separates inversion and Machine Learning training into two steps, this approach unifies both processes into a single pipeline. This reduces the computational burden of decoupled optimizations, allows increased flexibility to combine ML residual-minimization and CFD Jacobian inversion-based strategies, and finally ensures strict adherence to the Navier-Stokes equations. By propagating gradients through both the RANS equations and the Machine Learning model, the parameter space is directly constrained to respect physical laws. The differentiable BROADCAST solver [11] serves as the core CFD tool for this framework. This system wraps Python-compiled FORTRAN functions within a Machine Learning environment, enabling differentiation across solver iterations. The RANS layer, implemented in PyTorch [10], allows chain rule composition for operations before and after the application of the RANS equations. To predict the closing term in the RANS equations, a Deep Learning framework is applied. The strategy uses as input the mesh cell volumes and the flow's conservative state variables. Training and validation are conducted on a dataset specifically generated by systematic geometric and flow condition variations around the profile of the VKI LS-59 turbine blade [7]. The cases are generated via a Design of Experiments spanning different blade designs and flow configurations. Training data is obtained using the BROADCAST solver, adopting the Spalart-Allmaras turbulence model [12]. The purpose is to demonstrate the framework's capability to perform end-to-end optimization within a physically consistent pipeline.

1 Introduction

The design and development of aerodynamic profiles are crucial for determining the forces needed to size and manufacture components, especially in high-performance applications like turbines in jet engines. This process relies on advanced computational and experimental techniques to optimize performance. Computational Fluid Dynamics (CFD) simulations model airflow using turbulence models to solve the Reynolds-Averaged Navier-Stokes (RANS) equations, each model with different trade-offs between accuracy and computational cost. Accurate turbulence modeling is critical for predicting performance, heat transfer, and thermal stresses.

Designing aerodynamic profiles is iterative, beginning with theoretical data, followed by simulations and prototype testing to refine the design. Modern methods integrate optimization algorithms, such as Machine Learning (ML) [13, 14], to balance aerodynamics, thermodynamics, and structural mechanics, improving the overall design.

This work presents a novel approach to predicting turbulence models for RANS equations using Machine Learning techniques. The primary goal is to develop a system that enhances

aerodynamic simulations using Deep Learning (DL) techniques. That will be achieved with an end-to-end system [13] that leverages Convolutional Neural Networks (CNNs).
The proposed method utilizes the compressible Navier-Stokes BROADCAST solver [11], which uses Tapenade [5] for Algorithmic Differentiation (AD) [6]. This feature allows the exact calculation of the discrete adjoint operator, facilitating the implementation of adjoint optimization.
The pipeline is tested at first on the VKI LS-59 turbine blade test case [7] with transonic flow to perform an optimization on the eddy viscosity, and later on a dataset generated starting from the benchmark case to predict the eddy viscosity. This comprehensive approach aims to enhance the predictive accuracy of turbulence models, thereby contributing to more efficient and reliable aerodynamic simulations.

2 ML enhanced RANS

The stationary Reynolds-Averaged Navier-Stokes equations for compressible flows can be expressed as:

$$\begin{cases} \nabla\cdot(\rho\boldsymbol{U}) = 0 \\ \nabla\cdot(\rho\boldsymbol{U}\boldsymbol{U}) = -\nabla p + \nabla\cdot(\boldsymbol{\tau} + \boldsymbol{\tau}^R) \\ \nabla\cdot[(\rho E + p)\boldsymbol{U}] = \nabla\cdot\left((\boldsymbol{\tau} + \boldsymbol{\tau}^R)\cdot\boldsymbol{U} - \boldsymbol{q} - \boldsymbol{q_t}\right) \end{cases} . \tag{1}$$

To simplify the notations, the average quantities are not over-lined in these RANS equations: the density of the fluid ρ, the pressure p, the stress tensor due to viscosity $\boldsymbol{\tau}$ and the heat flux $\boldsymbol{q}$ are average quantities, whereas the velocity vector $\boldsymbol{U} = [U, V, W]$ and the total energy of the system E are mass-weighted average quantities. Finally, $\boldsymbol{\tau^R}$ and $\boldsymbol{q_t}$ are respectively the Reynolds stress tensor and the enthalpy turbulent diffusion flux.
To close the system, we need a model relating the Reynolds stress tensor $\boldsymbol{\tau^R}$ to the state variable $\boldsymbol{w} = [\rho, \rho U, \rho V, \rho W, \rho E]$. The most commonly implemented models make use of the Boussinesq eddy viscosity assumption [1], which states that the Reynolds stress tensor, τ_{ij}^R, is proportional to the trace-less mean strain rate tensor, S_{ij}, and to a scalar property called the eddy viscosity, μ_t, as shown in Equation (2):

$$\tau_{ij}^R = 2\mu_t\left(S_{ij} - \frac{1}{3}\frac{\partial U_k}{\partial x_k}\delta_{ij}\right). \tag{2}$$

The closure problem then transforms into finding a relation between the eddy viscosity and the state $\mu_t(\boldsymbol{w})$. In this work, the aim is finding this relationship with data and NNs rather than with a classical turbulence model (such as Spalart-Allmaras).
It is possible to write Equation (1) in compact form as:

$$\mathcal{R}_0(\boldsymbol{w}) + f(\boldsymbol{w}) = 0, \text{ with } f(\boldsymbol{w}) = \nabla\cdot\left[2\,\mu_t(\boldsymbol{w})\left(\boldsymbol{S} - \frac{1}{3}\nabla\cdot\boldsymbol{U}I\right)\right], \tag{3}$$

where $\mathcal{R}_0(\boldsymbol{w})$ designates the Navier-Stokes (NS) equations with just molecular stresses $\boldsymbol{\tau}$ and $\mu_t(\boldsymbol{w})$ is the function of interest to be parametrized, such as through the use of a NN.

3 Definition of the Optimization Problem

Given $\boldsymbol{w} \in \mathbb{R}^N$, where the number of degrees of freedom of the problem is $N = mn$, where m is the number of governing equations, while n is the number of cells discretizing the domain. Given the function $\mathcal{R}_0(\boldsymbol{w}) : \mathbb{R}^{mn} \to \mathbb{R}^{mn}$ that defines the the base-model, here the laminar NS equations, and $f_\vartheta(\boldsymbol{w}) : \mathbb{R}^{mn} \to \mathbb{R}^{mn}$ is a parametric version of function $f(\boldsymbol{w})$, through the eddy viscosity $\mu_{t,\vartheta}(\boldsymbol{w})$.
In the following, the full state variable $\boldsymbol{w}$ will be assumed known at the training stage. For example, this would be the case if a Direct Numerical Simulation (DNS) or Large Eddy Simulation (LES) solver was used to obtain the training data, since a straightforward time-averaging of such simulations is sufficient to reconstruct all variables within $\boldsymbol{w}$.
The cost function $\mathcal{J}$ to be minimized will be fixed as:

$$\mathcal{J}(\vartheta) = \|\mathcal{R}_0(\boldsymbol{w}) + f_\vartheta(\boldsymbol{w})\| + \alpha\|\mu_{t,\vartheta}(\boldsymbol{w})\|\,, \tag{4}$$

where the first term is the residual of the full RANS model and the latter is a regularization term that aims at maintaining the eddy viscosity small by tuning the parameter α.
This general notation remains valid for many different $\mu_{t,\vartheta}$. In this article two cases will be considered, which will be presented in Section 6:

- one could optimize w.r.t. the eddy viscosity, hence fixing:

$$\mu_{t,\vartheta}(\boldsymbol{w}) = \vartheta\,, \tag{5}$$

- one could optimize w.r.t. to an eddy viscosity model depending on the state $\boldsymbol{w}$, hence fixing:

$$\mu_{t,\vartheta}(\boldsymbol{w}) = \mathrm{NN}_\vartheta(\boldsymbol{w})\,, \tag{6}$$

  This model may be a CNN for example, being therefore able to find the best features in the flow state $\boldsymbol{w}$ to reproduce training data.

4 Differentiable Learning Pipeline

Due to its versatility and built-in functions, PyTorch [10] was selected to implement the ML routines in this work. The developed pipeline integrates one main module, **RANS**, with three inner layers, **BC**, **NS** and **FORCE** respectively to impose boundary conditions, impose the numerical scheme and compute the turbulent forcing of RANS. These inner routines work as interfaces with FORTRAN routines from BROADCAST. Indeed, every layer has to be differentiable, hence the operations of each layer must be overridden with the corresponding FORTRAN routine.
The **RANS** module applies boundary conditions to the state vector, then computes the numerical flux, hence the laminar residual, and finally, given f_ϑ, computes the turbulent forcing and adds it to the residual.

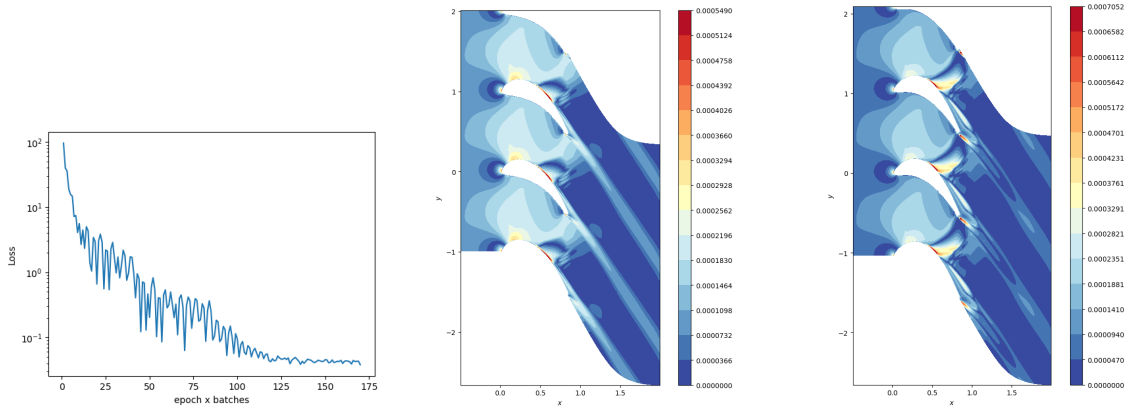
Table 1: Parameters and ranges used to generate the DOE.

Parameter	Minimal Value	Maximal Value
Distance of the 2° point from the 1° control point	-0.15	0.15
Angle of attack	10°	50°
Δx of the 3° control point from the original geometry	-0.2	0.2
Δy of the 3° control point from the original geometry	-0.2	0.2
Pitch (inter-blades distance)	0.68	1.1
Mach number at outflow	0.75	0.98

The custom inner layers override forward, backward, and Jacobian-vector product operations, facilitating seamless interaction with the external solver while ensuring differentiability.
The three internal layers can be concatenated inside a Pytorch module and Pytorch will then define the backward operation of the module accordingly.

5 Generation of the Dataset

First, to conduct a RANS simulation of a turbine cascade using the BROADCAST solver, several considerations must be made regarding mesh generation. The BROADCAST computational code employs a Finite Volume (FV) formulation with the high order Flux-Extrapolated MUSCL reconstruction described in [3]. This scheme is used on block-structured meshes to compute the necessary derivatives in the generalized i,j coordinates. Boundary conditions are enforced via ghost cells, with the number of ghost cells proportional to the chosen scheme's order. For the case of interest, specifically the VKI-LS59 profile [7], either a C-mesh or an H-mesh is commonly used in the literature. These mesh types allow for proper discretization of the boundary layer around the airfoil. Additionally, when using a deep learning model, special care must be taken, especially if spatial convolutions are required. To balance the requirements imposed by the solver, a single block-structured mesh discretizing the blade passage is considered. This approach splits the suction and pressure sides of the blade, positioning them relatively in the upper and lower parts of the computational domain. To achieve this, a custom elliptic mesh generator has been implemented. Specifically, by changing the control points used to define the camber line of the blade and by changing the angle of rotation of the blade, many different meshes can be computed with the same generator. By implementing the boundary conditions as the transonic case studied in [8], it was possible to compare the results with experimental data on the nominal case. The generated dataset can be found at [2].
To create the dataset a Design of Experiments (DOE) was generated using the maximum projection algorithm [9], Table 1 shows the intervals chosen for the parameters, details can be found at [2], and the following parameters were fixed for each simulation: (i) mesh sizes and geometry: $n_i = 300$, $n_j = 120$ ($n = n_i n_j$), geometry from DOE, (ii) scheme: FE-MUSCL of order 7, (iii) turbulence model: SA, (iv) physical parameters: $Mach_{in} = 0.275$, $Mach_{out}$ from DOE, α from DOE, $Re_{in} = 2.68 \times 10^5$, $Re_{out} = 6.8 \times 10^5$, $T_{ext} = 300\ K$, $P_{ext} = 1.01 \times 10^5\ Pa$.



(a) Behavior of the Loss function for 10 epochs, with a batch size of 50.

(b) Predicted eddy viscosity on a case with different geometry from the nominal one.

(c) Predicted eddy viscosity on a case with different geometry from the nominal one.

Figure 1: Results of the second POC on the generated dataset.

6 Proofs of Concept and Preliminary Results

Two proofs of concept (POCs) are performed with the developed routine.
The first, to validate the gradient concatenation, consists in an optimization on the eddy viscosity using only the nominal test case (turbine VKI LS-59). To do so a trainable parameter tensor of the dimension of the mesh is initialized with a constant value in the whole domain and is given as μ_t in the routine. The loss function is fixed as the sum of the norm of the residual of RANS (laminar plus the turbulent forcing) and a regularization term. Indeed, in order to perform this optimization the layer **RANS** is used.
By launching the routine for 1000 epochs (iterations), using as optimizer the Stochastic Gradient Descend (SGD), both the behavior of the loss function and the last obtained parameter field show the capabilities of this strategy. Indeed on this one case the routine is able to perfectly compute the eddy viscosity. Of course, since the state variables used were the result of the SA model, generated μ_t presents the same tails expected by the turbulent model.
As a second proof of concept a prediction on the developed dataset is studied. For this second study the loss function is fixed as described above in Section 3.In order to train a model to predict the turbulent viscosity a CNN was developed. Its architecture combines convolutions with varying kernel sizes, with the output excluding ghost cells. This design captures multi-scale features while preserving output consistency.
By launching BROADCAST routines in batch the whole routine can predict in batch. Using the SDG optimizer and fixing the number of epochs (iterations on the full dataset) to 10 and the batch size to 50, the results are presented in Figure 1. The norm of the turbulent residuals decrease significantly and the predicted eddy viscosity is consistent with the model used to generate the data.

7 Conclusion

This work demonstrated the implemented framework's ability to perform end-to-end optimization within a physically consistent pipeline. The optimization problem and dataset were introduced, and the routine was tested on two proofs of concept. The results confirm the method's effectiveness in predicting eddy viscosity from conservative flow variables, paving the way for further refinement and advancements in this field.

References

[1] J. Boussinesq. *Essai sur la théorie des eaux courantes*. Mémoires présentés par divers savants, Académie des Sciences, Paris, France, 1877.

[2] M. A. Bucci, L. Saverio, and F. Casenave. Vki-ls59: a 2d internal aero cfd rans dataset, under geometrical variations, Feb. 2025. URL https://doi.org/10.5281/zenodo.14840512.

[3] P. Cinnella and C. Content. High-order implicit residual smoothing time scheme for direct and large eddy simulations of compressible flows. *Journal of Computational Physics*, 326:1–29, 2016. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2016.08.023. URL https://www.sciencedirect.com/science/article/pii/S0021999116303734.

[4] L. Fang and P. He. Field inversion machine learning augmented turbulence modeling for time-accurate unsteady flow. *Physics of Fluids*, 36(5):055117, 05 2024. ISSN 1070-6631. doi: 10.1063/5.0207704. URL https://doi.org/10.1063/5.0207704.

[5] L. Hascoët. Tapenade: A tool for automatic differentiation of program. In *ECCOMAS 2004 - European Congress on Computational Methods in Applied Sciences and Engineering*, 2004.

[6] L. Hascoët. Automatic differentiation by program transformation. *INRIA Sophia-Antipolis, TROPICS team. Available on-line*, 2007.

[7] S. J. Hercus and P. Cinnella. Robust shape optimization of uncertain dense gas flows through a plane turbine cascade. In *ASME-JSME-KSME 2011 Joint Fluids Engineering Conference, AJK 2011*, volume 1, 2011. doi: 10.1115/AJK2011-05007.

[8] I. Jagodzińska, B. Olszański, K. Gumowski, and S. Kubacki. Experimental investigation of subsonic and transonic flows through a linear turbine cascade. *European Journal of Mechanics - B/Fluids*, 103: 182–192, 2024. ISSN 0997-7546. doi: https://doi.org/10.1016/j.euromechflu.2023.10.002. URL https://www.sciencedirect.com/science/article/pii/S0997754623001437.

[9] V. R. JOSEPH, E. GUL, and S. BA. Maximum projection designs for computer experiments. *Biometrika*, 102(2):371–380, 2015. ISSN 00063444. URL http://www.jstor.org/stable/43908541.

[10] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[11] A. Poulain, C. Content, D. Sipp, G. Rigas, and E. Garnier. Broadcast: A high-order compressible cfd toolbox for stability and sensitivity using algorithmic differentiation. *Computer Physics Communications*, 283:108557, 2023. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2022.108557. URL https://www.sciencedirect.com/science/article/pii/S0010465522002764.

[12] N. C. Rumsey. The Spalart-Allmaras Turbulence Model. https://turbmodels.larc.nasa.gov/spalart.html, 2020. Accessed: 2023-07-03.

[13] C. A. Ströfer and H. Xiao. End-to-end differentiable learning of turbulence models from indirect observations. *Theoretical and Applied Mechanics Letters*, 11, 2021. ISSN 20950349. doi: 10.1016/j.taml.2021.100280.

[14] J. L. Wu, H. Xiao, and E. Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 7, 2018. ISSN 2469990X. doi: 10.1103/PhysRevFluids.3.074602.