AMATH 590 - Literature Review: Turbulence Models - Application of Neural Networks and Reinforcement Learning

Deepak Venkatanarasimhan(1940141)

May 7, 2021

Abstract

This Literature review is written as a survey of turbulence and ML topics that help one understand the paper at `https://arxiv.org/abs/2005.09023`. I discuss Turbulence, fully-resolved solution methods (Direct Numerical Simulation (DNS)), steady-state solution methods (Reynolds Averaged Navier-Stokes (RANS)), and hybrid methods (Large Eddy Simulation (LES)). I also discuss the application of Neural Networks to closure models (Sub-grid scale models) and other models (correlation between wall shear stress and wall actuations) that may help in flow control. Finally, I discuss using Multi-agent Reinforcement Learning (MARL) to estimate sub-grid scale dynamics for an LES model.

1 Turbulence Models

The vast majority of naturally occurring flows are turbulent, so are many practical engineering flows. Hence the study of turbulence plays an important role in computational fluid dynamics. Although the physical nature of turbulence is not fully understood, one can obtain solutions numerically by setting up the Navier-Stokes with the correct initial, and boundary conditions [21]. The governing equations for Newtonian fluid flow - Navier-Stokes, as is well known is derived from the conservation laws of mass, momentum, and energy [17].

Even the simplest turbulent flows do not have analytical solutions. Hence, a complete description of a turbulent flow, where the flow variables are known as a function of space and time, can only be obtained by numerically solving the Navier-Stokes Equation [14]. Attempting to solve for the instantaneous flow-field creates computational challenges, except for low-Reynolds number flows with simple geometry. Turbulence is 3D (given that it is in the real world) and has an enormous range of scales to be resolved. Typically, the computational domain must be at least an order of magnitude larger than the scales characterizing the turbulence energy. At the same time, the mesh must be small enough to resolve eddies at Kolmogorov scales [21]. Hence, one resorts to modeling the statistical evolution of the flow field instead of solving for the entire instantaneous flow field. The key example of such a method is Reynolds-Averaged Navier-Stokes (RANS). It involves computing one-point moments such as mean velocity, or turbulent kinetic energy [14]. RANS solves for averaged quantities with the effects of all scales, i.e., it focuses on the steady-state flow. This leads to a significant reduction in computation time and makes RANS a backbone in industrial CFD applications. However, RANS does not capture the transient behavior of the flow. To retain transient details, at least for turbulent energy holding scales, one explores Large Eddy Simulation (LES). LES intermediates in complexity between DNS and RANS, and be retaining transient behavior is more accurate than RANS.

In LES, one separates motion into small and large scales by spatially filtering the velocity field with a kernel $G_\Delta(x)$ [13]. The LES equations are obtained by filtering the Navier-Stokes equations. This equation is amenable to discretization at a spatial resolution of order Δ, which is typically much more affordable than DNS (DNS requires resolutions near Kolmogorov scale). As explained above, a separation between resolved and unresolved scales is obtained by applying a spatial filtering operation. Navier-stokes governs the dynamics of the large, energy-carrying scales of motion. The sub-gride scale is modeled using one of many methods: i) Smagorinsky, ii) Dynamic Smagorinsky, iii) Similarity Models, etc. [13]. There are cases of flow where LES may or may not work. For momentum, heat, and mass transfer in free shear flows, there

is a cascade of energy, dominantly from resolved large scales to statistically isotropic & universally small scales. This is a great fluid for the use of LES [19]. LES may not be the right approach if the dynamics occur at small scales. The Navier-Stokes for the filtered velocity field $\tilde{u}$ is given by Equation 1. As one can see from the equation, we must compute the sub-grid scale stress tensor τ_{ij}^{Δ}. Alternately stated to close Equation 1, we must express the sub-grid scale stress tensor in terms of the filtered velocity field [13]. Please see Equation 3 for the sub-grid scale stress tensor. The unknown SGS stress tensor represents the effect of the SGS motions on the resolved fields of LES, which needs to be modeled using an SGS model so that the governing equations (specified below) can be solved [21].

$$\partial_t \tilde{u} + (\tilde{u}.\nabla)\tilde{u} = -1\frac{1}{p}\nabla\tilde{p} + v\nabla^2\tilde{u} - \nabla.\tau^{\Delta} \tag{1}$$

$$\nabla\tilde{u} = 0 \tag{2}$$

$$\tau_{ij}^{\Delta} = \widetilde{u_i u_j} - \tilde{u_i}\tilde{u_j} \tag{3}$$

The classic smagorinsky model is a linear eddy viscosity model that relates the SGS stress tensor to the filtered rate of strain; please see Equation 4. v_t is the turbulent eddy viscosity, and $\tilde{S}$ is the large-scale stress tensor, and Δ is the filter/grid size [8].

$$\tau^{\Delta} - \frac{1}{3}tr(\tau^{\Delta}) = -2v_t\overline{S} \tag{4}$$

$$v_t = (C_s\Delta)^2\left|\overline{S}\right| \tag{5}$$

$$\left|\overline{S}\right| = (2\overline{S_{ij}S_{ij}})^{\frac{1}{2}} \tag{6}$$

$$\overline{S_{ij}} = \frac{1}{2}(\frac{\partial\tilde{u_i}}{\partial x_j} + \frac{\partial\tilde{u_j}}{\partial x_i}) \tag{7}$$

Since small scales tend to be more isotropic than large ones, it should be possible to parameterize them using simpler and more universal models than the standard Reynolds stress model. Thus most sub-grid scale models are based on the eddy-viscosity assumption given by Equation 5. This assumes that the small scales are in equilibrium, i.e., energy production and dissipation are in balance. The smagorinsky constant $C_s = 0.2$ gives good results for fine grid resolutions. However, the smagorinsky constant may need to be 0 during the early and nonlinear stages of transition. To better reflect the local state of flow, one uses dynamic SGS that exhibits proper asymptotic behavior near solid boundaries or in laminar flow without requiring damping or intermittency functions [6]. Expressions for the large scale strain rate tensor $\tilde{S}$ in terms of large scale velocity $\tilde{u}$ is given by Equations 6 and 7. Please also see [18] for some more discussion on LES models of turbulent flow.

For a more detailed overview of Turbulent flows, one can refer to the books at [15], and [16]. In the next section, I discuss the concept of Closure models and how ML is used to model some of these equations from data.

2 Closure Models and ML

Governing equations derived from first principles need to be augmented by some form of closure term, which typically incorporates information from different physical effects or scales [1]. For instance, in RANS, when the Navier-stokes equations are averaged, they become unclosed. In other words, there are more variables than equations. Closure models are empirical formulae that increase the number of equations to match the number of unknowns [12]. RANS models have closures to represent the turbulent stresses and scalar fluxes emerging from the averaging process. More details on RANS closure models are at [4].

In this paper, I have focussed mainly on LES. Hence, I discuss the method employed to capture an SGS model described in [5]. SGS modeling can be computationally expensive. Computing the turbulent viscosity coefficient C_s is the core of a turbulent flow numerical simulation. It is related to the local conditions of the flow, making it computationally expensive. One can train an NN to predict C_s. [5] shows that it saves 20% time compared to the complete dynamic procedure, and the quality is good compared with a DNS solution. For input, we use streamwise, wall-normal, and spanwise components of velocity and instantaneous fluctuations. The choice of input vector was dictated by the idea of introducing all the variables involved in the calculation of the original SGS model and to let the NN perform a sensitivity analysis and proper pruning of the unnecessary inputs [5]. With this paper, once again, we can show that NN thrives in situations where an unknown functional relationship exists between input and output, and enough data is available to approximate this function.

3 Control with Neural Networks

The ability to control turbulent flow is of significant economic interest. Successful control of turbulent boundary layers by reducing drag can result in a substantial reduction in operational cost for, say, aircraft and marine vehicles [2].

In turbulent channel flows, the near-wall region contains the most fundamental mechanisms for the generation of turbulence. The near-wall region is characterized by coherent vortical structures, manifested by alternating low and high-speed streamwise wall streaks [9]. In practice, designing a control scheme that measures important quantities requires thought. In the case of [2], the NN input consists of wall shear stresses, and the output is wall actuation (wall-normal velocity). About seven neighboring points in the spanwise direction and one in the streamwise direction provide enough information to train and control near-wall structures. Output from the network is used as input to the actuator. Based on optimal structure from offline training, they implement an online inverse model controller in numerical experiments of a turbulent channel flow. It reduces drag by 20%.

[9] compares linear POD and NN as is often done to capture coherent vortical structures in the near-wall region. Linear POD is a subset of a more general family of nonlinear transformations implemented by a neural network. These architectures can be viewed as compression models for the near-wall flow field. They require the entire field as input, producing a compressed version described by the POD or nonlinear PCA components. In the context of flow control, it is desirable to reconstruct the flow field in the near-wall region using wall-only information such as wall pressure and shear stresses. Such reconstruction is essential to devise successful control schemes. For instance, as in the prior paper [2], a neural network is used to learn the correlation between wall shear stresses and wall actuation to implement opposition control in a turbulent channel flow.

It is also possible to imagine NN models being useful in constructing near-wall modes for flow solvers using RANS or LES to get computational performance gains [9].

4 Brief review of Reinforcement Learning

Reinforcement Learning (RL) is learning what to do - how to map situations to actions - to maximize a numerical reward signal [20]. RL uses agents that perform actions. The agents learn the mapping from action to rewards, but they must also explore all possible activities. On a stochastic task, the agent must try each action repeatedly to gain a reliable estimate of the expected reward. RL tackles the whole problem instead of a subproblem and is a bridge between other engineering disciplines and ML [20]. To use RL successfully in situations approaching real-world complexity, agents require sensory inputs and must use these to generalize past experience to new situations [11]. RL appears to have achieved some success in domains that can be fully observed from low-dimensional state spaces. The well-known Alpha Go example where 'value networks' are used to evaluate board positions and 'policy networks' to select moves was very successful. One can read about the performance of the machine against a human player at [3].

RL agents interact with the environment and collect information in an offline setting called Batch RL. This is in contrast to the online setting where data becomes available in sequential order and is used to progressively update the behavior of the agent [10]. RL can be formalized as a discrete-time stochastic control process. Agent starts at $s_0 \in S$, by gathering an initial observation $w_0 \in \Omega$. At each time step t, agent takes action $a_t \in A$. It observes a reward $r_t \in R$. RL is specified as a Markov decision process so that only the most recent state matters. The goal of an RL agent is to find a policy $\pi(s, a) \in \Pi$ so as to optimize a Q function that captures expected return or reward (See Equation 8 to 10). For a more detailed discussion, please see [10].

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma Q^{\pi}(s', a = \pi(s'))) \tag{8}$$

$$Q^{*}(s,a) = \max_{\pi \in \Pi} Q^{\pi}(s,a) \tag{9}$$

$$\pi^{*}(s) = \operatorname*{argmax}_{a \in A} Q^{*}(s,a) \tag{10}$$

In fluid mechanics, one is confronted with nonlinear problems of high dimensionality [7]. This makes the state and action space large for any control application. RL could be an effective way to specify the overall optimal control problem. For examples see [7]. Alternately, one could set up RL to compute closure terms in an automated manner, as we'll see in the next section.

5 MARL paper overview

In place of using supervised learning to model closure terms, the paper at [8] explores multi-agent reinforcement learning (MARL) as an automated discovery tool of turbulence models. The paper uses this approach on LES models of isotropic turbulence. The agents receive a reward for closing the gap to a DNS solution of the system. The closure model is a control policy enacted by cooperating agents, which detect spatiotemporal patterns in the flow field to estimate the unresolved subgrid-scale physics.

A quick rehash on LES as discussed in prior sections - LES models operate at two scales: i) one end of the spectrum we encounter the integral scales, which depend on the specific forcing, flow geometry, or boundary conditions, ii) at the other end are the Kolmogorov scales at which turbulent kinetic energy is dissipated. Closure models describe terms at the Kolmogorov scales. In the last 50 years, SGS models have been constructed using physical insight, numerical approximations, and often problem-specific intuition [8]. Recent advances in hardware and algorithms have fueled a broad interest in the development of data-driven turbulence models [4].

Error computation in supervised learning approaches often rely on one-step target values for the model (e.g., SGS stresses computed from filtered DNS). Hence, the resultant NN model is not trained to compensate for the evolution of discrepancies between DNS and LES data and the compounding errors. The paper addresses these challenges by controlling under-resolved simulations by locally adapting the coefficients of the Smagorinsky closure model to accurately reproduce the energy spectrum predicted by DNS [8]. Since a cumulative reward guides the RL strategy over time, it overcomes compounding modeling errors. By using multiple agents, adaptability is improved to localized flow features. The paper benchmarks the MARL model against Classical Smagorinsky and Dynamic Smagorinsky models. It is observed that MARL models minimize error over all future steps, avoid growing energy buildup and numerical instability at higher Reynolds Numbers. Hence, we can say that MARL maximizes high-level objectives computed from direct application of the learned model and produces SGS models that are stable under perturbation and resistant to compounding errors. MARL minimizes the discrepancies between the energy spectra of LES and that computed from orders of magnitude more computationally expensive, fully resolved simulations (DNS) [8].

The results are promising, albeit one makes the simplifying assumption of isotropic turbulence to minimize the computational cost of DNS. Hence future iterations with more realistic fluid-flows may be challenging. It is known that many turbulent flows are not stationary, homogeneous, or isotropic. It may be prohibitively

expensive to measure statistical properties for such flows through multiple realizations of DNS [8]. The paper shares that it is possible to train without expensive DNS. Energy spectra, wall shear stresses, or drag coefficients from experiments can train SGS models.

6 Summary and Outlook

The prediction of the statistical properties of turbulent flows is critical for engineering (cars to nuclear reactors), science (ocean dynamics to astrophysics), and government policy (climate and weather forecasting) [8]. In this literature review, we study concepts related to Turbulent Flows and understand existing PDE-based solutions such as DNS, RANS, and LES that mainly use physics insight. We know the computational intractability of DNS for complex fluids and the limitations of RANS in understanding transient behavior. LES appears to be a good compromise for flows, provided the majority of the rate-controlling processes occur in the large resolved scales. When using modeling for smaller/ Kolmogorov scales, one encounters closure terms that account for the impact of unresolved scales. These closure models again can be derived via physics insight or in a data-driven manner using Machine Learning. We show the application of Neural Networks to estimate sub-grid scale model terms. We also explore a simple control prototype that reduces drag by modeling near-wall region flows. Finally, we look at MARL as an approach to estimate a sub-grid scale model for LES on an isotropic turbulent fluid. This is an improvement over prior supervised SGS models discussed in [5] and improves performance and stability as discussed in [8].

Using ML models to improve the computational performance of turbulence closure modeling or perform control tasks is an active area of research. One continues to verify the results for numerical accuracy, stability, and performance for fluids with different properties (higher Reynolds numbers, nonstationarity, etc.) and types of flow cavities/ surfaces. Improvements in understanding and performance gain are likely to have immediate commercial application in CFD solvers or in performing control tasks.

References

[1] Andrea Beck et. al. *Deep Neural Networks for Data-Driven LES Closure Models*. 2019.

[2] Changhoon Lee et. al. *Application of Neural Network to Turbulence Control for Drag Reduction*. 1997.

[3] David Silver et. al. *Mastering the game of Go with deep neural networks and tree search*. 2016.

[4] Duraisamy et. al. *Turbulence Modeling in the Age of Data*. 2019.

[5] F. Sarghini et. al. *Neural networks based subgrid scale modeling in large eddy simulations*. 2001.

[6] Massimo Germano et. al. *A dynamic subgrid-scale eddy viscosity model*. 1990.

[7] Paul Garnier et. al. *A Review of Deep Reinforcement Learning for Fluid Mechanics*. 2021.

[8] Petros Koumoutsakos et. al. *Automating Turbulence Modeling by Multi-Agent Reinforcement Learning*. 2020.

[9] Petros Koumoutsakos et. al. *Neural Network Modeling for Near Wall Turbulent Flow*. 2002.

[10] Vincent François-Lavet et. al. *An Introduction to Deep Reinforcement Learning*. Now, 2018.

[11] Volodymyr Mnih et. al. *Human-level control through deep reinforcement learning*. 2015.

[12] Paul A. Durbin. *Some Recent Developments in Turbulence Closure Modeling*. 2018.

[13] Charles Meneveau and Joseph Katz. *Scale-invariance and Turbulence Models for Large-Eddy Simulation*. 2000.

[14] Parviz Moin and Krishnan Mahesh. *Direct Numerical Simulation: A Tool in Turbulence Research*. 1998.

[15] S.B.Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[16] P. Sagaut. *Large Eddy Simulation for Incompressible Flows*. Springer, 2006.

[17] Alexander Smits. *Viscous Flows and Turbulence*. 2009.

[18] Stephen.B.Pope. *Simple models of turbulent flows*. 2011.

[19] Stephen.B.Pope. *Ten questions concerning the large-eddy simulation of turbulent flows*. 2004.

[20] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2012.

[21] Yang Zhiyin. *Large-eddy simulation: Past, present and the future*. 2014.