Enhancing computational fluid dynamics with machine learning

Ricardo Vinuesa1,2 and Steven L. Brunton3

Machine learning is rapidly becoming a core technology for scientific computing, with numerous opportunities to advance the field of computational fluid dynamics. Here we highlight some of the areas of highest potential impact, including to accelerate direct numerical simulations, to improve turbulence closure modeling and to develop enhanced reduced-order models. We also discuss emerging areas of machine learning that are promising for computational fluid dynamics, as well as some potential limitations that should be taken into account.

The field of numerical simulation of fluid flows is generally known as computational fluid dynamics (CFD). Fluid mechanics is an area of great importance, both from a scientific perspective and for a range of industrial-engineering applications. Fluid flows are governed by the Navier–Stokes equations, which are partial differential equations (PDEs) modeling the conservation of mass and momentum in a Newtonian fluid. These PDEs are nonlinear due to the convective acceleration (which is related to the change of velocity with the position), and they commonly exhibit time-dependent chaotic behavior, known as turbulence.

Solving the Navier–Stokes equations for turbulent flows requires numerical methods that may be computationally expensive, or even intractable at high Reynolds numbers (Re), due to the wide range of scales in space and time necessary to resolve these flows. There are various approaches to solve these equations numerically, which can be discretized using methods of different orders, for instance finite-difference1, finite-volume2, finite-element3, spectral methods4. There are various approaches to solve these equations numerically, which can be discretized using methods of different orders, for instance finite-difference1, finite-volume2, finite-element3, spectral methods4. A similar approach was developed by Stevens and Colonius22 to improve the results of fifth-order finite-difference schemes in the context of shock-capturing simulations. Jeon and Kim23 proposed to use a deep neural network to simulate the well-known finite-volume discretization scheme employed in fluid simulations. They tested their method with reactive flows, obtaining very good agreement with reference high-resolution data, but at one-tenth the computational cost. However, they also documented errors with respect to the reference solution, which increased with time. Another deep-learning approach, based on a fully convolutional/long-short-term-memory (LSTM) network, was proposed by Stevens and Colonius24 to improve the accuracy of finite-difference/finite-volume methods. Second, we consider the strategy of developing a correction between

Accelerating direct numerical simulations

Direct numerical simulation (DNS) is a high-fidelity approach in which the governing Navier–Stokes equations are discretized and integrated in time with enough degrees of freedom to resolve all flow structures. Turbulent flows exhibit a pronounced multi-scale character, with vortical structures across a range of sizes and energetic content. This complexity requires fine meshes and accurate computational methods to avoid distorting the underlying physics with numerical artifacts. With a properly designed DNS, it is possible to obtain a representation of the flow field with the highest level of detail of the CFD methods. However, the fine computational meshes required to resolve the smallest scales lead to exceedingly high computational costs, which increase with the Reynolds number.

A number of ML approaches have been developed recently to improve the efficiency of DNS. We first discuss several studies aimed at improving discretization schemes. Bar-Sinaï et al.21 proposed a technique based on deep learning to estimate spatial derivatives in low-resolution grids, outperforming standard finite-difference methods. A similar approach was developed by Stevens and Colonius22 to improve the results of fifth-order finite-difference schemes in the context of shock-capturing simulations. Jeon and Kim23 proposed to use a deep neural network to simulate the well-known finite-volume discretization scheme employed in fluid simulations. They tested their method with reactive flows, obtaining very good agreement with reference high-resolution data, but at one-tenth the computational cost. However, they also documented errors with respect to the reference solution, which increased with time. Another deep-learning approach, based on a fully convolutional/long-short-term-memory (LSTM) network, was proposed by Stevens and Colonius24 to improve the accuracy of finite-difference/finite-volume methods.

In this Perspective we focus on the potential of ML to improve CFD, including possibilities to increase the speed of high-fidelity simulations, develop turbulence models with different levels of fidelity, and produce reduced-order models beyond what can be achieved with classical approaches. Several authors have surveyed the potential of ML to improve fluid mechanics, including topics beyond the scope of CFD, such as experimental techniques, control applications and related fields. Others have reviewed specific aspects of ML for CFD, such as turbulence closure13,14 and heat-transfer aspects of CFD for aerodynamic optimization15. Our discussion will address the middle ground of ML for CFD more broadly, with a schematic representation of topics covered in Fig. 1. Approaches to improve CFD with ML are aligned with the larger efforts to incorporate ML into scientific computing, for example via physics-informed neural networks (PINNs)16,17 or to accelerate computational chemistry8,18.

1FLOW, Engineering Mechanics, KTH Royal Institute of Technology, Stockholm, Sweden. 2Swedish e-Science Research Centre (SeRC), Stockholm, Sweden.
3Department of Mechanical Engineering, University of Washington, Seattle, WA, USA. E-mail: rvinuesa@mech.kth.se
It is also possible to accelerate CFD by solving the Poisson equation with deep learning. This has been done by several research groups in various areas, such as simulations of electric fields and particles. The Poisson equation is frequently used in operator-splitting methods to discretize the Navier–Stokes equations, where first the velocity field is advected, and the resulting field $\mathbf{u}^*$ does not satisfy the continuity equation (that is, for incompressible flows, $\mathbf{u}^*$ does not satisfy the divergence-free condition). The second step involves a correction to ensure that $\mathbf{u}^*$ is divergence-free, leading to the following Poisson equation:

$$\frac{\Delta t}{\rho} \nabla^2 p = -\nabla \cdot \mathbf{u}^*, \quad (1)$$

where $\Delta t$ is the simulation time step, $\rho$ is the fluid density and $p$ is the pressure. Solving this equation is typically the most computationally expensive step of the numerical solver. Therefore, devising alternate strategies to solve it more efficiently is an area of great promise. ML can be used for this task, because this technology can exploit data from previous examples to find mappings between the divergence of the uncorrected velocity and the resulting pressure field. For instance, Ajuria et al. proposed using a convolutional neural network (CNN) to solve equation (1) in incompressible simulations, and tested it in a plume configuration. Their results indicate that it is possible to outperform the traditional Jacobi solver with good accuracy at low Richardson numbers $\text{Ri}$ (the Richardson number measures the ratio between the buoyancy and the shear in the flow). However, the accuracy degrades at higher $\text{Ri}$, motivating the authors to combine the CNN with the Jacobi solver (which is used when the CNN diverges). CNNs were also used to solve the Poisson problem by decomposing the original problem into a homogeneous Poisson problem plus four inhomogeneous Laplace subproblems. This decomposition resulted in errors below 10%, which motivates using this approach as a first guess in iterative algorithms, potentially reducing computational cost. Other data-driven methods have been proposed to accelerate the pressure correction in multi-grid solvers. These approaches may also be used to accelerate simulations of lower fidelity that rely on turbulence models.

Numerical simulations can also be accelerated by decreasing the size of the computational domain needed to retain the physical properties of the system. Two ways in which the domain can be reduced are to replace a section upstream of the domain of interest by an inflow condition, and to replace part of the far-field region by a suitable boundary condition. In doing so, these parts of the domain do not need to be simulated, thus producing computational savings, and ML can help to develop the inflow and far-field conditions as discussed next. Fukami et al. developed a time-dependent inflow generator for wall-bounded turbulence simulations using a convolutional autoencoder with a multilayer perceptron (MLP). They tested their method in a turbulent channel flow at $\text{Re}_T = 180$, which is the friction Reynolds number based on channel half height and friction velocity, and they maintained turbulence for an interval long enough to obtain converged turbulence statistics. This is a promising research direction due to the fact that current inflow-generation methods show limitations in terms of generality of the inflow conditions, for instance at various flow geometries and Reynolds numbers. A second approach to reduce the computational domain in external flows is to devise a strategy to set the right pressure-gradient distribution without having to simulate the far field. This was addressed by Morita et al. through Bayesian optimization based on Gaussian-process regression, achieving accurate results when imposing concrete pressure-gradient conditions on a turbulent boundary layer.

**Improving turbulence models**

DNS is impractical for many real-world applications because of the computational cost associated with resolving all scales for flows with high Reynolds numbers, together with difficulties arising from complex geometries. Industrial CFD typically relies on either Reynolds-averaged Navier–Stokes (RANS) models, where no turbulent scales are simulated, or coarsely resolved large-eddy simulations (LES), where only the largest turbulent scales are resolved and smaller ones are modeled. Here the term model refers to an a priori assumption regarding the physics of a certain range of turbulent scales. In the following we discuss ML applications to RANS and LES modeling.

**RANS modeling.** Numerical simulations based on RANS models rely on the so-called RANS equations, which are obtained after decomposing the instantaneous flow quantities into a mean and a...
fluctuating component, and averaging the Navier–Stokes equations in time. Using index notation, the instantaneous $i$th velocity component $\bar{u}_i$ can be decomposed into its mean ($U_i$) and fluctuating ($u_i$) components as follows: $\bar{u}_i = U_i + u_i$. Although the RANS equations govern the mean flow, the velocity fluctuations are also present in the form of the Reynolds stresses $\overline{u_i u_j}$ (where the overbar denotes averaging in time), which are correlations between the $i$th and $j$th velocity components. Because it is convenient to derive equations containing only mean quantities, $\overline{u_i u_j}$ needs to be expressed as a function of the mean flow, and this is called the closure problem. The first approach to do this was proposed by Boussinesq$^{42}$, who related the Reynolds stresses to the mean flow via the so-called eddy viscosity, $\nu_T$. Although this approach has led to some success, there are still a number of open challenges for RANS modeling in complex flows$^{43}$, where this approach is too simple. In particular, conventional RANS models exhibit notable errors when dealing with complex pressure-gradient distributions, complicated geometries involving curvature, separated flows, flows with a substantial degree of anisotropy and 3D effects, among others. As argued by Kutz$^{44}$, ML can produce more sophisticated models for the Reynolds stresses by using adequate data, in particular if the examples used for training represent a sufficiently rich set of flow configurations.

A wide range of ML methods have recently been used to improve RANS turbulence modeling$^{45}$, focusing on the challenge of improving the accuracy of the Reynolds stresses for general conditions. For example, Ling et al.$^{45}$ proposed a novel architecture, including a multiplicative layer with an invariant tensor basis, used to embed Galilean invariance in the predicted anisotropy tensor. Incorporating this invariance improves the performance of the network, which outperforms traditional RANS models based on linear$^{46}$ and nonlinear$^{47}$ eddy-viscosity models. They tested their models for turbulent duct flow and the flow over a wavy wall, which are challenging to predict with RANS models$^{48}$ because of the presence of secondary flows$^{49}$. Other ML-based approaches rely on physics-informed random forests to improve RANS models, with applications to cases with secondary flows and separation. On the other hand, Jiang et al.$^{50}$ recently developed an interpretable framework for RANS modeling based on a physics-informed residual network (PiResNet). Their approach relies on two modules to infer the structural and parametric representations of turbulence physics, and includes non-unique mappings, a realizability limit and noise-insensitivity constraints. Interpretable models are essential for engineering and physics, particularly in the context of turbulence modeling. The interpretability of the framework by Jiang et al.$^{51}$ relies on its constrained model form$^{52}$, although this is not generally possible to attain. Recent studies$^{53}$ have discussed various approaches to include interpretability in the development of deep-learning models, and one promising approach is the one proposed by Cranmer and others$^{54}$. This technique has potential in terms of embedding physical laws and improving our understanding of such phenomena. Other interpretable RANS models were proposed by Weatheritt and Sandberg$^{55}$, using gene-expression programming (GEP), which is a branch of evolutionary computing. GEP iteratively improves a population of candidate solutions by survival of the fittest, with the advantage of producing closed-form models. The Reynolds-stress anisotropy tensor was also modeled by Weatheritt and Sandberg$^{56}$, who performed tests in RANS simulations of turbulent ducts. Models based on sparse identification of nonlinear dynamics (SINDy)$^{57}$ have also been used for RANS closure models$^{58}$. Furthermore, the literature also reflects the importance of imposing physical constraints in the models and incorporating uncertainty quantification (UQ)$^{59}$ alongside ML-based models. It is also important to note that, when using DNS quantities to replace terms in the RANS closure, the predictions may be unsatisfactory$^{60}$. This inadequacy is due to the strict assumptions associated with the RANS model, as well as the potential ill-conditioning of the RANS equations$^{61}$. It is thus essential to take advantage of novel data-driven methods, while also ensuring that uncertainties are identified and quantified. Another interesting review by Ahmed et al.$^{62}$ discussed both classical and emerging data-driven closure approaches, also connecting with reduced-order models (ROMs).

Obiols-Sales et al.$^{63}$ developed a method to accelerate the convergence of RANS simulations based on the very popular Spalart–Allmaras turbulence model$^{64}$ using the CFD code OpenFOAM$^{65}$. In essence, they combined iterations from the CFD solver and evaluation of a CNN model, obtaining a convergence that was from 1.9 to 7.4 times faster than that of the CFD solver, both in laminar and turbulent flows. Multiphase flows, which consist of flows with two or more thermodynamic phases, are also industrially relevant.

**Fig. 2** | An example of ML-accelerated direct numerical simulation. a. Results from the work by Kochkov et al.$^{23}$, where the instantaneous vorticity field is shown for simulations with high/low resolution, as well as for a case with low resolution supplemented with ML. Four different time steps are shown, and some key vortical structures are highlighted with yellow squares. Adapted from ref. $^{25}$, with permission from the United States National Academy of Sciences. b. Sketch showing that ML accelerates the simulation through a correction between the coarse and fine resolutions. In this example one can reduce the resolution by downsampling the flow data on a coarser mesh, and then use ML to recover the details present in the finer simulation through super-resolution. The correction between coarse and fine resolutions, which is based on ML, enables the super-resolution task here.
Gibou et al.\textsuperscript{71} proposed different directions in which ML and deep learning can improve the CFD of multiphase flows, in particular when it comes to enhancing the simulation speed. Ma et al.\textsuperscript{7} used deep learning to predict the closure terms (that is, gas flux and streaming stresses) in their two-fluid bubble flow, whereas Mi et al.\textsuperscript{7} analyzed gas–liquid flows and employed NNs to identify the different flow regimes.

**LES modeling.** LES-based numerical simulations rely on low-pass filtering the Navier–Stokes equations (Fig. 1c), such that the largest scales are simulated and the smallest ones (below a certain cutoff) are modeled by means of a so-called subgrid-scale model (SGS). Note that the smallest scales are the most demanding from a computational point of view (both in terms of computer time and memory usage), because they require fine meshes to be properly resolved. The first proponent of this type of approach was Smagorinski\textsuperscript{74}, who developed an SGS model based on an eddy viscosity, which was computed in terms of the mean flow and the grid size. His model assumed that the production equals the dissipation for small scales. Although LES can lead to substantial computational savings with respect to DNS while exhibiting good accuracy, there are still challenges associated with its usage for general purposes\textsuperscript{63}. For example, current SGS models exhibit limited accuracy in their predictions of turbulent flows at high Reynolds numbers, in complex geometries and in cases with shocks and chemical reactions.

ML has also been used to develop SGS models in the context of LES of turbulent flows in basically two ways: supplementing the unresolved energy in the coarse mesh using supervised learning and developing agent-based approaches to stabilize the coarse simulation. When it comes to the first approach, in the following we list several studies that rely on high-fidelity data to train both deep-learning and GEP-based models. First, Beck et al.\textsuperscript{75} used an artificial NN based on local convolutional filters to predict the mapping between the flow in a coarse simulation and the closure terms, using a filtered DNS of decaying homogeneous isotropic turbulence. Lapeyre et al.\textsuperscript{89} employed a similar approach, with a CNN architecture inspired by a U-net model, to predict the subgrid-scale wrinkling of the flame surface in premixed turbulent combustion. They obtained better results than classical algebraic models. Maulik et al.\textsuperscript{91} employed an MLP to predict the SGS model in an LES using high-fidelity numerical data to train the model. They evaluated the performance of their method on Kraichnan turbulence\textsuperscript{92}, which is a classical 2D decaying-turbulence test case. Several other studies have also used NNs in a supervised manner for SGS modeling\textsuperscript{93–96}. Furthermore, GEP has been used for SGS modeling\textsuperscript{82} in a LES of a Taylor–Green vortex, where it outperformed standard LES models. Regarding the second approach to LES modeling, Novati et al.\textsuperscript{91} employed multi-agent reinforcement learning (RL) to estimate the unresolved subgrid-scale physics. This unsupervised method exhibits favorable generalization properties across grid sizes and flow conditions, and the results are presented for isotropic turbulence. As shown in the schematic representation of Fig. 3, the state of the agents at a particular instant is given in terms of both local and global variables. This state is then used to calculate the so-called dissipation coefficient at each grid point.

In certain applications, for example those involving atmospheric boundary layers, the Reynolds number is several orders of magnitude larger than those of most studies based on turbulence models or wind-tunnel experiments\textsuperscript{97}. The mean flow in the inertial sublayer has been widely studied in the atmospheric boundary layer community, and it is known that in neutral conditions it can be described by a logarithmic law\textsuperscript{98}. The logarithmic description of the inertial sublayer led to the use of wall models, which replace the region very close to the wall with a model defining a surface shear stress matching the logarithmic behavior. This is the cornerstone of most atmospheric models, which avoid resolving the computationally expensive scales close to the wall. One example is the work by Giometto et al.\textsuperscript{99}, who studied a real urban geometry, adopting the LES model by Bou-Zeid et al.\textsuperscript{100} and the Moeng model\textsuperscript{101} for the wall boundary condition. It is possible to use data-driven approaches to develop mappings between the information in the outer region (which is resolved) and a suitable off-wall boundary condition (so the near-wall region does not need to be resolved). For example, it is possible to exploit properties of the logarithmic layer and rescale the flow in the outer region to set the off-wall boundary condition in turbulent channels\textsuperscript{102,103}. This may also be accomplished via transfer functions in spectral space\textsuperscript{104}, CNNs\textsuperscript{105} or modeling the temporal dynamics of the near-wall region via deep NNS\textsuperscript{106}. Other promising approaches based on deep learning are the one by Moriya et al.\textsuperscript{107}, based on defining a virtual velocity, and the RL technique by Bae and Koumoutsakos\textsuperscript{108}. Defining off-wall boundary conditions with ML is a challenging yet promising area of research.
Reduced-order models are typically designed to balance efficiency and accuracy. ML solutions further improve the efficiency by reducing the effective dimension of the model and increasing the accuracy through better modeling of how these few variables co-evolve in time. In this figure, the input is a high-resolution flow field evolving in time (f0), and the output is a reconstruction of that field from the latent space. a, Classic POD/PCA may be viewed as a shallow autoencoder with a single encoder \( U \) and decoder \( V \) layers, together with linear activation units. For the flow past a cylinder example, as shown, the dynamics evolve in a 3D coordinate system. b, A deep, multi-level autoencoder with multilayer encoder \( \phi \) and decoder \( \psi \), as well as nonlinear activation functions, provides enhanced nonlinear coordinates on a manifold. The cylinder flow now evolves on a 2D submanifold. c, The classic Galerkin projection model, obtained by projecting the governing Navier–Stokes equations onto an orthogonal basis. d, The Galerkin projection model in c can be replaced by more generic ML regressions, such as LSTM networks, reservoir networks or sparse nonlinear models, to represent the nonlinear dynamical system \( \hat{z} = f(z) \).

Developing reduced-order models
ML is also being used to develop ROMs in fluid dynamics. ROMs rely on the fact that even complex flows often exhibit a few dominant coherent structures that may provide coarse, but valuable information about the flow. Thus, ROMs describe the evolution of these coherent structures, providing a lower-dimensional, lower-fidelity characterization of the fluid. In this way, ROMs provide a fast surrogate model for the more expensive CFD techniques described above, enabling optimization and control tasks that rely on many model iterations or fast model predictions. The cost of this efficiency is a loss of generality: ROMs are tailored to a specific flow on many model iterations or fast model predictions. The cost of this

![Fig. 4 | Schematic of NN autoencoders for dimensionality reduction and model identification. Reduced-order models are typically designed to balance efficiency and accuracy. ML solutions further improve the efficiency by reducing the effective dimension of the model and increasing the accuracy through better modeling of how these few variables co-evolve in time. In this figure, the input is a high-resolution flow field evolving in time (f0), and the output is a reconstruction of that field from the latent space. a, Classic POD/PCA may be viewed as a shallow autoencoder with a single encoder \( U \) and decoder \( V \) layers, together with linear activation units. For the flow past a cylinder example, as shown, the dynamics evolve in a 3D coordinate system. b, A deep, multi-level autoencoder with multilayer encoder \( \phi \) and decoder \( \psi \), as well as nonlinear activation functions, provides enhanced nonlinear coordinates on a manifold. The cylinder flow now evolves on a 2D submanifold. c, The classic Galerkin projection model, obtained by projecting the governing Navier–Stokes equations onto an orthogonal basis. d, The Galerkin projection model in c can be replaced by more generic ML regressions, such as LSTM networks, reservoir networks or sparse nonlinear models, to represent the nonlinear dynamical system \( \hat{z} = f(z) \).](image-url)
fitting the observed dynamics to the fewest terms in a library of candidate functions that might describe the dynamics, resulting in models that are interpretable and balance accuracy and efficiency. SINDy has been used to model a range of fluids, including laminar and turbulent wake flows, convective flows and shear flows.

The modeling approaches above may be combined with a deep autoencoder to uncover a low-dimensional latent space, as in the SINDy autoencoder. DMD has also been extended to nonlinear coordinate embeddings through the use of deep-autoencoder networks. In all of these architectures, there is a tremendous opportunity to embed partial knowledge of the physics, such as conservation laws, symmetries and invariances. It may also be possible to directly impose stability in the learning pipeline.

The ultimate goal for ML ROMs is to develop models that have improved accuracy and efficiency, better generalizability to new initial and boundary conditions, flow configurations, varying parameters, as well as improved model interpretability, ideally with less intrusive methods and less data. Enforcing partially known physics, such as symmetries and other invariances, along with sparsity, is expected to be critical in these efforts. It is also important to continue integrating these efforts with the downstream applications of control and optimization. Finally, many applications of fluid dynamics involve safety-critical systems, and therefore certifiable models are essential.

### Emerging possibilities and outlook

In this Perspective we have provided our view on the potential of ML to advance the capabilities of CFD, focusing on three main areas: accelerating simulations, enhancing turbulence models and improving reduced-order models. In Table 1 we highlight some examples of applications within each of the three areas. There are also several emerging areas of ML that are promising for CFD, which we discuss in this section. One area is non-intrusive sensing—that is, the possibility of performing flow predictions based on, for example, information at the wall. This task, which has important implications for closed-loop flow control, has been carried out via CNNs in turbulent channels. In connection to the work by Guastoni et al., there are a number of studies documenting the possibility of performing super-resolution predictions (for example, when limited flow information is available) in wall-bounded turbulence using CNNs, autoencoders and generative-adversarial networks. Another promising direction is the imposition of constraints based on physical invariances and symmetries on the ML model, which has been used for SGS modeling, ROMs and for geophysical flows.

PINNs constitute another family of methods that are becoming widely adopted for scientific computing, more broadly. This technique, introduced by Raissi et al., uses deep learning to solve PDEs, exploiting the concept of automatic differentiation used in the back-propagation algorithm to calculate partial derivatives and form the equations, which are enforced through a loss function. Although the method is different from the ML approaches discussed above, in certain cases PINNs provide a promising alternate approach to traditional numerical methods for solving PDEs. This framework also shows promise for biomedical simulations, in particular after the recent work by Raissi et al., in which the concentration field of a tracer is used as an input to accurately predict the instantaneous velocity fields by minimizing the residual of the Navier–Stokes equations. The growing usage of PINNs highlights the relevance of exploiting the knowledge we have about the governing equations when conducting flow simulations. In fact, PINNs are being used for turbulence modeling (solving the RANS equations without the need of a model for the Reynolds stresses), for developing ROMs, for dealing with noisy data and for accelerating traditional solvers, for example, by means of solving the Poisson equation more efficiently.

| Table 1 | Summary of applications of ML to enhance CFD |
|---------------------------------------------|---------------------------------------------|
| **Accelerate DNS** | **Improve LES/ROMS** | **Develop ROMs/ML** | **Future developments** |
| Deep learning for correction between coarse and fine simulations | Deep learning/RL for more general subgrid-scale models | POD for improved linear subspaces to represent dynamics | Deep learning for prediction and control of turbulent flows |
| Deep learning for accelerating the solution of the Poisson equation | Deep learning/RL for more robust wall models | Autoencoders to learn manifolds to represent dynamics | Better interpretability of deep learning for CFD models |
| Domain reduction: deep learning for inflow generation; Gaussian processes for pressure-gradient distributions | Deep learning/symbolic regression for more accurate and general RANS models | Deep learning for representing temporal system dynamics | More efficient operation of high-performance computing facilities facilitated by ML |
| Deep learning for improved accuracy of finite-difference/finite-volume schemes | Deep learning/symbolic regression for interpretable RANS models | Sparse identification of nonlinear dynamics for learning efficient and interpretable ROMs | ML-enabled more advanced computational architectures |

Shown are examples of approaches to accelerate DNSs, improvement of turbulence models for LES and RANS simulations, development of ROMs and possible future developments.

There are also grand challenges in CFD that necessitate new methods in ML. One motivating challenge in CFD is to perform accurate coarse-resolution simulations in unforced 3D wall-bounded turbulent flows. The production of turbulent kinetic energy (TKE) in these flows takes place close to the wall (although at very high Reynolds numbers, outer-layer production also becomes relevant), and therefore using coarse meshes may substantially distort TKE production. In these flows of high technological importance, the TKE production sustains the turbulent fluctuations, so a correction between coarse and fine resolutions may not be sufficient to obtain accurate results. Another challenge is the need for robust and interpretable models, a goal that is not easily attainable with deep-learning methods. There are, however, promising directions to achieve interpretable deep-learning models, with important applications to CFD. A clear challenge of CFD is the large energy consumption associated with large-scale simulations. In this sense, ML can be an enabler for more efficient operation of the high-performance computing (HPC) facilities, or even for future computational architectures, including quantum computers, as shown in Table 1.

It is worth noticing that, although ML has very high potential for CFD, there are also a number of caveats that may limit the applicability of ML to certain areas of CFD. First, ML methods, such as deep learning, are often expensive to train and require large amounts of data. It is therefore important to identify areas where ML outperforms classical methods, which have been established for decades, and may be more accurate and efficient in certain cases. For example, it is possible to develop interpretable ROMs with traditional methods such as POD and DMD, and although deep learning can provide some advantages, the simpler ML methods may be more efficient and straightforward. Finally, it is important to...
assess the information about the training data available to the user: certain flow properties (for example, incompressibility, periodicity and so on) should be embedded in the ML model to increase training efficiency and prediction accuracy. There is also a question of how the training data are generated, and whether the associated cost is taken into account when benchmarking. In this context, transfer learning is a promising area\(^3\).

Despite the caveats above, we believe that the trend of advancing CFD with ML will continue in the future. This progress will continue to be driven by an increasing availability of high-quality data, high-performance computing and a better understanding and facility with these emerging techniques. Improved adoption of reproducible research standards\(^4,5\) is also a necessary step. Given the critical importance of data when developing ML models, we advocate that the community continue to establish proper benchmark systems and best practices for open-source data and software so as to harness the full potential of ML to improve CFD.

Received: 28 January 2022; Accepted: 17 May 2022;
Published online: 27 June 2022

References

1. Godunov, S. & Bobachevsky, I. A finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. Mat. Sb. 47, 271–306 (1959).
2. Eymard, R., Gallouët, T. & Herbin, R. Finite volume methods. Handb. Numer. Anal. 7, 713–1018 (2000).
3. Zienkiewicz, O. C., Taylor, R. L., Nithiarasu, P. & Zhu, J. Z. The Finite Element Method, 3 (Elsevier, 1977).
4. Canuto, C., Hussaini, M. Y., Quarteroni, A. & Zang, T. A. Spectral Methods in Fluid Dynamics (Springer Science & Business Media, 2012).
5. Brunton, S. L. & Kutz, J. N. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control (Cambridge Univ. Press, 2019).
6. Recht, B. A tour of reinforcement learning: the view from continuous systems. Mach. Learn. (2017 IEEE International Conference on Advanced Packaging and Systems Technology (EASAP) 1–3 (IEEE, 2017).
7. Vinuesa, R. et al. The role of artificial intelligence in achieving the Sustainable Development Goals. Nat. Commun. 11, 133 (2020).
8. Noé, F., Tkatchenko, A., Müller, K.-R. & Clementi, C. Machine learning for molecular simulation. Annu. Rev. Phys. Chem. 71, 361–390 (2020).
9. Niederer, S. A., Sacks, M. S., Girolami, M. & Willcox, K. Scaling digital twins from the artisanal to the industrial. Nat. Comput. Sci. 1, 313–320 (2021).
10. Samuel, A. L. Some studies in machine learning using the game of checkers. IBM J. Res. Dev. 3, 210–229 (1959).
11. Brenner, M., Eldredge, J. & Freund, J. Perspective on machine learning for advancing fluid mechanics. Phys. Rev. Fluids 4, 100501 (2019).
12. Brunton, S. L., Noack, B. R. & Kouroumtakis, P. Machine learning for fluid mechanics. Annu. Rev. Fluid Mech. 52, 477–508 (2020).
13. Duraisamy, K., Iaccarino, G. & Xiao, H. Turbulence modeling in the age of data. Annu. Rev. Fluid Mech. 51, 357–377 (2019).
14. Ahmed, S. E. et al. On closures for reduced order models—A spectrum of first-principle to machine-learned avenues. Phys. Fluids 33, 091301 (2021).
15. Wang, B. & Wang, J. Application of artificial intelligence in computational fluid dynamics. Ind. Eng. Chem. Res. 60, 2772–2790 (2021).
16. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707 (2019).
17. Karniadakis, G. E. et al. Physics-informed machine learning. Nat. Rev. Phys. 3, 422–440 (2021).
18. Noé, F., Olsson, S., Köhler, J. & Wu, H. Boltzmann generators: sampling in turbulent systems. Proc. Natl. Acad. Sci. USA 118, e2101784118 (2021).
19. Chandler, G. J. & Kerswell, R. R. Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow. J. Fluid Mech. 722, 554–595 (2013).
20. Bauer, P., Thorpe, A. & Brunet, G. The quiet revolution of numerical weather prediction. Nature 525, 47–55 (2015).
21. Schenk, F. et al. Warm summers during the Younger Dryas cold reversal. Nat. Commun. 9, 1638 (2018).
22. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine learning for fluid dynamics. Theor. Comput. Fluid Dyn. 34, 483–496 (2020).
23. Jeon, J., Lee, J. & Kim, S. I. Finite volume method network for the acceleration of unsteady computational fluid dynamics: Non-reacting and reacting flows. Int. J. Energy Res. https://doi.org/10.1002/er.7879 (2022).
82. Reissmann, M., Hasslbergerb, J., Sandberg, R. D. & Klein, M. Application of
direct numerical simulation of turbulence. J. Comput. Phys. 325, 22–37 (2016).

81. Maulik, R. & San, O. A neural network approach for the blind
detection of turbulent flows. J. Fluid Mech. 831, 151–187 (2017).

80. Reissmann, M., Hasslberger, J., Sandberg, R. D. & Klein, M. Application of
generation to four-dimensional LES model of a Taylor Green vortex. J. Comput. Phys. 424, 109859 (2021).
113. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporal chaotic systems from data: a reservoir computing approach. Phys. Rev. Lett. 120, 024102 (2018).

114. Kaiser, E. et al. Cluster-based reduced-order modelling of a mixing layer. J. Fluid Mech. 754, 365–414 (2014).

115. Peherstorfer, B. & Willcox, K. Data-driven operator inference for nonintrusive projection-based model reduction. Comput. Meth. Appl. Mech. Eng. 306, 196–215 (2016).

116. Benner, P., Goyal, P., Kramer, B., Peherstorfer, B. & Willcox, K. Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. Comput. Meth. Appl. Mech. Eng. 372, 113433 (2020).

117. Qian, E., Kramer, B., Peherstorfer, B. & Willcox, K. Lift & Learn: physics-informed machine learning for large-scale nonlinear dynamical systems. Physica D 406, 132401 (2020).

118. Loiseau, J.-C. & Brunton, S. L. Constrained sparse Galerkin regression. J. Fluid Mech. 838, 42–67 (2018).

119. Loiseau, J.-C. Data-driven modeling of the chaotic thermal convection in an annular thermosyphon. Theor. Comput. Fluid Dyn. 34, 339–365 (2020).

120. Guan, Y., Brunton, S. L. & Novoselov, I. Sparse nonlinear models of chaotic electroconvection. R. Soc. Open. Sci. 8, 202367 (2021).

121. Deng, N., Noack, B. R., Morzynski, M. & Pastur, L. R. Low-order model for successive bifurcations of the fluidic pinball. J. Fluid Mech. 884, A37 (2020).

122. Champion, K., Lusch, B., Kutz, J. N. & Brunton, S. L. Galerkin force model for transient and post-transient dynamics of the fluidic pinball. J. Fluid Mech. 918, A4 (2021).

123. Callaham, J. L., Rigas, G., Loiseau, J.-C. & Brunton, S. L. An empirical mean-field model of symmetry-breaking in a turbulent wake. Sci. Adv. 8, eabm4786 (2022).

124. Callaham, J. L., Brunton, S. L. & Loiseau, J.-C. On the role of nonlinear correlations in reduced-order modelling. J. Fluid Mech. 938, A1 (2022).

125. Champion, K., Lusch, B., Kutz, J. N. & Brunton, S. L. Data-driven discovery of coordinates and governing equations. Proc. Natl. Acad. Sci. USA 116, 22445–22451 (2019).

126. Yeung, E., Kundu, S. & Hodas, N. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. Preprint at https://arxiv.org/abs/1708.06850 (2017).

127. Takeshi, N., Kawahara, Y. & Yairi, T. Learning Koopman invariant subspaces for dynamic mode decomposition. In Advances in Neural Information Processing Systems 1130–1140 (ACM, 2017).

128. Lusch, B., Kutz, J. N. & Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. Nat. Commun. 9, 4950 (2018).

129. Mardt, A., PasquaI, L., Wu, H. & No', F. V AMPnets: deep learning of embeddings of nonlinear dynamics. Nat. Commun. 5, 5 (2018).

130. Otto, S. E. & Rowley, C. W. Linearly-recurrent autoencoder networks for transient and post-transient dynamics of the fluidic pinball. J. Fluid Mech. 918, A4 (2021).

131. Fukami, K., Fukagata, K. & Taira, K. Super-resolution reconstruction of turbulent flows with machine learning. J. Fluid Mech. 870, 106–120 (2019).

132. Liu, X. & Vinuesa, R. Physics-informed neural network reduced order model with shallow polynomial nonlinear terms. J. Comput. Phys. 451, 110841 (2021).

133. Vinuesa, R., Lehmkuhl, O., Lozano-Durán, A. & Rabault, J. Fluid control in wings and discovery of novel approaches via deep reinforcement learning. Fluids 6, 281–302 (2019).

134. Guastoni, L. et al. Convolutional-network models to predict wall-bounded turbulence from wall quantities. J. Fluid Mech. 928, A27 (2021).

135. Kim, H., Kim, J., Won, S. & Lee, C. Unsupervised deep learning for super-resolution reconstruction of turbulence. J. Fluid Mech. 910, A29 (2021).

136. Vinuesa, R., Fukagata, K., Kawai, K., & Taira, K. Physics-informed machine learning for large-scale nonlinear dynamical systems. Theor. Comput. Fluid Dyn. 33, 884–910 (2019).

137. Kim, J., Moin, P. & Moser, R. Turbulence statistics in fully developed channel flow at low Reynolds number. J. Fluid Mech. 177, 133–166 (1987).

138. Fukami, K., Fukagata, K. & Taira, K. Towards quantum computing of turbulence. Nat. Comput. Sci. 2, 68–69 (2022).

139. Barbera, L. A. The hard road to reproducibility. Science 354, 142–146 (2016).

140. Mesnard, O. & Barba, L. A. Reproducible and replicable computational fluid dynamics: it’s harder than you think. Comput. Sci. Eng. 19, 44–55 (2017).

141. Fukami, K., Nakamura, T. & Fukagata, K. Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. Phys. Fluids 32, 091110 (2020).

142. Riess, M., Yazdani, A. & Karniadakis, G. E. Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. Science 367, 1026–1030 (2020).

143. Evazi, H., Tahani, M., Schlatter, P. & Vinuesa, R. Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations. Preprint at https://arxiv.org/abs/2107.10711 (2021).

144. Kim, Y., Choy, Y., Widemann, D. & Zohdi, T. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. J. Comput. Phys. 451, 110841 (2021).

145. Evazi, H. & Vinuesa, R. Physics-informed deep-learning applications to experimental fluid mechanics. Preprint at https://arxiv.org/abs/2203.15402 (2022).

146. Marksiek, L. The old and the new: can physics-informed deep-learning replace traditional linear solvers? Front. Big Data https://doi.org/10.3389/fdata.2021.669097 (2021).

147. Kim, J., Moin, P. & Moser, R. Turbulence statistics in fully developed channel flow at low Reynolds number. J. Fluid Mech. 177, 133–166 (1987).

148. Fukagata, K. Towards quantum computing of turbulence. Nat. Comput. Sci. 2, 68–69 (2022).

149. Barba, L. A. The hard road to reproducibility. Science 354, 142–146 (2016).

150. Mesnard, O. & Barba, L. A. Reproducible and replicable computational fluid dynamics: it’s harder than you think. Comput. Sci. Eng. 19, 44–55 (2017).

Acknowledgements

R.V. acknowledges financial support from the Swedish Research Council (VR) and from ERC grant no. 2021-CoG-101043998, DEEPCONTROL. S.L.B. acknowledges funding support from the Army Research Office (ARO W911NF-19-1-0045; programme manager M. Munson).

Author contributions

Both authors contributed equally to the ideation of the study and the writing process.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence should be addressed to Ricardo Vinuesa. Peer review information Nature Computational Science thanks Michael Brenner and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Principal Handling Editor: Jie Pan, in collaboration with the Nature Computational Science team.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations. © Springer Nature America, Inc. 2022

NATURE COMPUTATIONAL SCIENCE