Creating data-driven CFD workflows using OpenFOAM and PyTorch

Andre Weiner[1], Chiara Pesci[2], Tomislav Marić[3], Dieter Bothe[4]

[1]*Technical University of Braunschweig, Institute of Fluid Mechanics, Braunschweig, Germany, a.weiner@tu-braunschweig.de*

[2]*Engineering System International GmbH, Neu-Isenburg, Germany, chiara.pesci@esi-group.com*

[3]*Technical University of Darmstadt, Mathematical Modeling and Analysis, Darmstadt, Germany, maric@mma.tu-darmstadt.de*

[4]*Technical University of Darmstadt, Mathematical Modeling and Analysis, Darmstadt, Germany, bothe@mma.tu-darmstadt.de*

1. Computational fluid dynamics and machine learning

Computational fluid dynamics (CFD) simulations are an integral part of the work of engineers in industry and academia. In recent years, supplementing CFD simulations with data from different sources has seen rising popularity and demand. However, the idea of incoporating data into simulations is not new at all. Instead, it is a necessity. The numerical solution of transport equations requires the knowledge of material properties like viscosity, molecular diffusivity, or thermal conductivity. In combustion simulations involving tens or hundreds of chemical species, it is a common practice to create and use look-up tables for reaction properties. In the field of turbulence modeling, there is no model that does not rely on a multitude of empirical coefficients or blending functions that are often determined based on data coming from high-fidelity simulations or experiments. The genuinely new character of recent developments lies in the complexity of data-based models and the extent to which data is harvested.

Data can come from a variety of sources, and it can come in a multitude of forms. CFD simulations create data as they compute and write field or surface data for velocity, pressure, or temperature. Moreover, simulations also create secondary runtime data like residuals, required iterations, adjusted time steps, memory usage,

or parallel efficiency. Sometimes, information is only available implicitly in the form of an environment[1] that we can manipulate and observe. All of the aforementioned data may be used to create data-driven CFD workflows with the ultimate goal of creating more accurate, more efficient models. The field concerned with the creation of mathematical models based on sample data is called machine learning (ML).

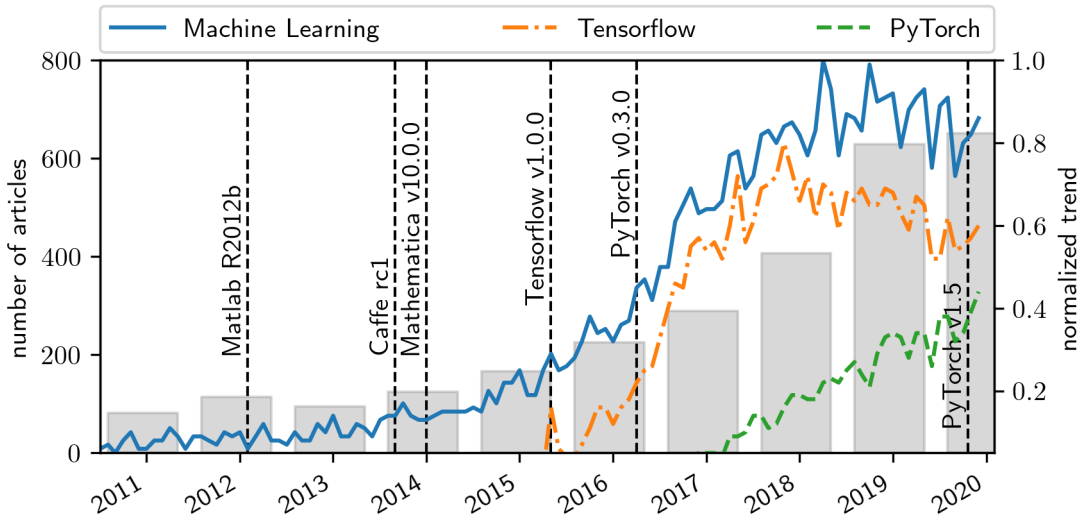

Figure 1: Popularity of the search terms "machine learning", "Tensorflow", and "PyTorch" over the last nine years according to Google trends (https://trends.google.de/trends/). The number of articles per year corresponds to the number of search results on ScienceDirect for the query "computational fluid dynamics and machine learning". The dashed vertical lines indicate software releases introducing noteworthy support for deep learning.

Several factors have led to the explosion of interest in ML and CFD over the past years. Firstly, the strongly increasing compute power and the vast amounts of complex data demand similarly sophisticated tools to analyze and leverage such data. Secondly, figure 1 indicates a strong correlation between software releases with significant ML support according to nowadays standards, the popularity of ML in general, and the academic throughput in terms of research articles dealing with the

1 For the meaning of evironment in the context of reinforcement learning refer to https://en.wikipedia.org/wiki/Reinforcement_learning

combination of CFD and ML. Note that the number of articles in figure 1 is certainly too high and may be seen as an upper bound. However, more fine-grained studies show a similar trend; see chapter 2 in [1]. Especially the open-source libraries Tensorflow and PyTorch have become extremely popular due to their ease of use and feature-richness.

Coupling CFD and ML models may be categorized according to the three main branches of ML: supervised, unsupervised, and reinforcement learning. An overview of the present literature body in these categories may be found in [1] for CFD and ML, in [2] for fluid dynamics in general and ML, and in [3] specifically for fluid dynamics and reinforcement learning. In the following, we list potential applications of each learning type in CFD. The examples are intentionally detailed and diverse with the hope to spark ideas in the reader's mind for utilization in her or his field of interest.

Supervised learning is undoubtedly the field with the largest potential to create models with immediate impact. Typical ML algorithms used in unsupervised learning are neural networks, decision trees, or polynomials. Supervised learning may be used to create

- wall functions for LES or RANS simulations based on DNS or experimental data,
- more realistic boundary conditions for velocity and other fields based on prior simulations or experimental data,
- a classification model to infer the stability regime of particle trajectories,
- a curvature model that computes the curvature based on the volume fraction field in a volume-of-fluid simulation with surface tension,
- rheological models based on simulation approaches operating below the continuum-mechanical scale or rheometer data,
- improved correlations for lift, drag, mass or heat transfer, enhancement, or effective reaction kinetics in Euler-Lagrange-type simulations.

Unsupervised learning deals with two kinds of problems, namely, dimensionality reduction and clustering. Typical algorithms employed for dimensionality reduction are proper orthogonal decomposition (PCA), dynamic mode decomposition (DMD), or

auto-encoders. Frequently used clustering algorithms are K-means or Gaussian mixture models. Unsupervised learning may be used to

- find coherent structures in turbulent flows,
- create reduced-order models for the transient drag acting on a car,
- find faulty simulations in parameter studies with hundreds or thousands of parameter variations,
- identify conditions in which a solver performs particularly well or bad.

Among the three main learning types, reinforcement learning is probably the one promising the most astonishing and impactful models. Combined with CFD simulations, reinforcement learning can be used to

- reduce the drag acting on a car by means of active flow control,
- decrease the load on an airfoil by active flow control,
- perform direct shape optimization to enhance the performance of a heat exchanger,
- adjust solution control parameters of a simulation to accelerate the numerical solution.

For some of the examples mentioned before, the reader might have a good idea of how to create and use a data-driven model. For other examples, the concept might be rather vague or completely alien. Even if the mathematical modeling idea is clear, there might still be some lack of knowledge about the exact set-up and implementation. This contribution aims to clarify the workflow by providing start-to-end examples for different scenarios, including software requirements, data processing, data analysis, as well as model creation, assessment, and deployment.

2. OpenFOAM and PyTorch

PyTorch is a library specialized in neural networks that was first released in 2016 by Facebook. However, the roots of PyTorch date back to the early 2000s when the Torch library was initially released. The primary purpose of both libraries is to provide the user with N-dimensional arrays and array operations that can be performed on different processing units like CPUs and GPUs. Another characteristic that both libraries share is the separation into backend and frontend. The backend

implements low-level operations, while the frontend facilitates the interaction with the library. In contrast to its predecessor, the backend of PyTorch is mostly implemented in C++ instead of C, and Python has replaced Lua as the main frontend language. Since version 1.5, PyTorch also has a stable C++ frontend, making it an ideal candiate to integrate supervised or unsupervised ML models into OpenFOAM solvers. Moreover, OpenFOAM's run time selection mechanism and PyTorch's neural network module provide an ideal environment for the implementation of reinforcement learning applications. Considering that Python is currently the language of choice for data analysis and visualization, PyTorch also comes with built-in support to create models via Python and to deploy them in C++ environments. Another benefit of combining PyTorch and OpenFOAM is the possibility to perform mixed-precision operations on CPUs and GPUs at run time. This ability may be used to exploit CPU and GPU, for example, to offload post-processing tasks to the GPU.

3. Examples for data-driven CFD workflows

The primary value in this contribution lies in the detailed start-to-end examples, including step-by-step guidance and source code examples. First, we provide instructions to set-up a suitable environment for jointly using OpenFOAM and PyTorch. Second, an example of supervised learning from [1] to create high-fidelity reference data is examined in detail. Thereafter, we touch on examples for each learning type:

- The so-called high-Schmidt number problem makes it extremely challenging to resolve species concentration boundary layers. Insufficient mesh resolution may lead to unphysical results or to numerical instabilities if the concentration is coupled to the fluid flow [5, 6]. We present an ML-based model to approximate the boundary layer with high accuracy even if the boundary layer is much smaller than the cell size [4].

- Raw turbulent flow data may be hard to comprehend visually due to the chaotic nature of turbulence. We demonstrate how to employ POD to shed some light into the darkness.

- The coupling of pressure and velocity is a challenging step in the solution of the momentum equation. The challenge is even greater in multiphase flow solvers like interface [5, 6, 7] or front-tracking approaches [8] and results in long run times or numerical instabilities. We explore how a reinforcement learning agent may be used to adjust solution control parameters.

Finally, we provide an overview of best practices and an outlook on upcoming applications.

Acknowledgements

Andre Weiner and Dieter Bothe thank the German Research Foundation (DFG) for funding received within the priority program SPP1740 “Reactive bubbly flows” under the grant BO1879/13-2.

References

[1] A. Weiner: *Modeling and simulation of convection-dominated species transfer at rising bubbles*, Ph.D. Thesis, TU Darmstadt, 2020

[2] S. L. Brunton, B. R. Noack, P. Koumoutsakos: *Machine learning for fluid danamics, Annual Review of Fluid Mechanics*, 52, 477-508, 2020

[3] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, E. Hackem: *A review on deep reinforcement learning for fluid mechanics*, arXiv:1908.04127, 2019

[4] A. Weiner, D. Hillenbrand, H. Marschall, D. Bothe: *Data-driven subgrid-scale modeling for convection-dominated concentration boundary layers*, Chemical Engineering & Technology, 42 (7), 1349-1356, 2019

[5] C. Pesci: *Computational analysis of fluid interfaces influenced by soluble surfactant*, Ph.D. Thesis, TU Darmstadt, 2019

[6] C. Pesci, A. Weiner, H. Marschall, D. Bothe: *Computational analysis of single rising bubbles influenced by soluble surfactant*, Journal of Fluid Mechanics, 856, 209-763, 2018

[7] Z. Tuković, H. Jasak: *A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow*, Computers & Fluids, 55, 70-84, 2012

[8] T. Tolle, T. Marić, D. Bothe: *SAAMPLE: A segregated accuracy-driven algorithm for multiphase pressure-linked equations*, Computers & Fluids, 200, 104450, 2020