

PERFORM: A Python package for developing reduced-order models for reacting fluid flows

Christopher R. Wentland¹ and Karthik Duraisamy¹

¹ Department of Aerospace Engineering, University of Michigan

DOI: [10.21105/joss.03428](https://doi.org/10.21105/joss.03428)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Kyle Niemeyer](#) ↗

Reviewers:

- [@clemaitre58](#)
- [@Himscipy](#)

Submitted: 13 April 2021

Published: 06 December 2021

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Combusting fluid flows are common in a variety of engineering systems, driving the gas turbines that generate our electricity, providing the thrust for rocket engines, and powering our cars and airplanes. However, the use of computational fluid dynamics (CFD) to simulate these phenomena is practically impossible for time-critical, many-query applications such as parametric design, failure prediction, and uncertainty quantification. Data-driven reduced-order models (ROMs) have shown potential for vastly reducing the computational cost of evaluating CFD models. In general, ROMs learn a low-dimensional representation of the high-dimensional system state (which may be as many as tens or hundreds of millions of degrees of freedom) and evolve this low-dimensional state in time at a much lower computational cost. Research on applying ROMs to practical reacting flows is currently in its early stages, and initial results have shown that standard ROM techniques may be ineffective for this class of problems ([Huang et al., 2018](#); [Huang & Duraisamy, 2019](#)). The dearth of research on this topic may be attributed to the complexity of reacting flow modeling combined with a lack of approachable open-source libraries for combustion CFD.

The Prototyping Environment for Reacting Flow Order Reduction Methods (PERFORM) is a Python package designed to allow rapid implementation, testing, and evaluation of ROMs for one-dimensional reacting flows. It combines a robust compressible reacting flow solver with a modular framework for deploying new ROM methods. This eliminates much of the software development difficulty for ROM researchers who may have little experience with combustion modeling or low-level programming languages, allowing them to perform research with a challenging, practical class of problems.

Statement of need

The ROM community spans many scientific disciplines, and efforts to build and release tools for ROM research and applications have been commensurately broad. Some open-source projects provide implementations of standard ROM methods within a specific flow solver ([Grunloh et al., 2021](#); [Hess & Rozza, 2020](#); [Hesthaven et al., 2016](#); [Stabile et al., 2017](#); [Stabile & Rozza, 2018](#)). Other libraries have been developed to provide generic interfaces with dynamical system solvers and supply ROM capabilities, most notably Pressio ([Rizzi et al., 2020](#)) and pyMOR ([Milk et al., 2016](#)). These libraries allow CFD practitioners to test a suite of standard ROM methods with an existing CFD solver. However, these libraries may not be useful to ROM method developers, who may find it difficult and prohibitively time-consuming to implement new ROM methods in a very complex software environment or in a low-level programming language. Further, those who wish to test ROM methods on combustion problems may not have access to a robust and efficient combustion CFD solver (many are closed-source) or the know-how to use general open-source solvers such as OpenFOAM.

PERFORM aims to ease some of these difficulties by packaging a one-dimensional compressible reacting flow solver with a flexible object-oriented ROM framework. It trades flexibility of application for flexibility of ROM development, focusing on a single class of problems while making the code extremely accessible to ROM researchers of all backgrounds. For one, PERFORM is written purely in Python, a widely-used, easily-learned, and flexible high-level programming language. Additionally, ROMs in PERFORM are structured within a class hierarchy which maximizes code re-use and minimizes developer effort in implementing variations of ROM methods.

In most cases, only a basic understanding of Python classes and NumPy/SciPy operations is required to implement new ROM methods. Further, all elements of the underlying reacting flow solver are separated from the ROM classes, ensuring that ROM method developers do not need to interact with the solver routines. As such, the learning curve for understanding PERFORM's ROM mechanics and implementing new ROM methods is relatively gentle. Note that although pyMOR (Milk et al., 2016) similarly provides a pure Python ROM framework, it aims to be far more generalizable to any class of problems, be compatible with a host of low-level computational and data handling libraries, and enable parallel computing. While this makes pyMOR much more flexible and efficient, it can also make ROM prototyping and code maintenance much more cumbersome.

PERFORM's public repository comes with several benchmark cases which are ready to run out-of-the-box. As will be discussed below, these benchmark cases address several of the critical issues facing the broader ROM community, particularly the difficulty of propagating transient flow features beyond the training data set and making accurate predictions in a complex parameter space. We hope that by providing these benchmark cases, the community can measure and compare ROM methods for a more challenging, yet manageable, class of problems.

Features

PERFORM is designed with modularity and expansion in mind. Generic class interfaces are provided to allow for the simple implementation of new gas models, reaction models, flux schemes, time integration schemes, and boundary conditions. Generic class interfaces are also provided for projection-based ROMs, linear trial spaces, and non-linear trial manifolds via autoencoders. Beyond this, the solver comes with many standard accessories, such as restart files, live visualizations of field and probe monitor data, and several useful pre-/post-processing scripts.

At the time of submitting this paper, PERFORM is specifically equipped with the following features:

- Solver
 - calorically-perfect gas model
 - irreversible finite-rate reaction model
 - Roe flux difference scheme (Roe, 1981)
 - several explicit Runge-Kutta time integration schemes
 - implicit BDF time integration schemes, with or without dual time-stepping (Venkateswaran & Merkle, 1995)
- ROMs
 - linear Galerkin (explicit and implicit) (Rowley et al., 2004), LSPG (Carlberg et al., 2017), and MP-LSVT (Huang et al., 2021) projection ROMs with gappy POD hyper-reduction (Everson & Sirovich, 1995)

- 87 – non-linear Galerkin (explicit and implicit), LSPG (Lee & Carlberg, 2020), and
- 88 MP-LSVT projection ROMs via autoencoders
- 89 – generic interface to Keras (Chollet & others, 2015) neural network models

90 We are working to provide generic class interfaces for non-intrusive ROM methods and ROM
 91 stabilization methods (e.g. closure, filtering, artificial viscosity). We are also implementing a
 92 thermally-perfect gas model, reversible finite rate reaction model, and a generic interface for
 93 PyTorch neural network models.

94 Applications

95 PERFORM was created within an ongoing US Air Force Office of Scientific Research Center of
 96 Excellence collaboration between researchers from the University of Michigan, the University
 97 of Texas at Austin, New York University, and Purdue University investigating ROMs for rocket
 98 combustors. Many team members do not have intimate experience with combustion modeling,
 99 and PERFORM was originally developed for internal use by those members to test novel ROM
 100 methods on simplified combustor flows. Research using PERFORM in the development of
 101 robust linearized ROMs has already been published (Rezaian et al., 2021). Further work has
 102 begun in using it to perform basis and sampling adaptation for hyper-reduction, as well as to
 103 compute premixed flamelet manifolds.

104 Recent community-wide discussions have begun encouraging ROM researchers to pursue more
 105 practical fluid flow applications. This was synthesized in the “Data-driven Modeling for Com-
 106 plex Fluid Physics” panel at the 2021 AIAA SciTech Forum, where leaders in the ROM
 107 community discussed a general need to evaluate ROMs beyond the traditional “toy” problems
 108 (e.g. 1D Burgers’, 2D lid-driven cavity). PERFORM was presented as a companion code to
 109 this workshop, and provides several benchmark cases for use in future meetings of the work-
 110 shop. These benchmark cases include a Sod shock tube, a transient multi-species contact
 111 surface (with and without artificial acoustic forcing), a stationary premixed flame (with arti-
 112 ficial acoustic forcing), and a transient premixed flame (with and without artificial acoustic
 113 forcing). These benchmarks are intended to provide some cohesion for the ROM community
 114 to address more complex, practical systems.

115 Acknowledgements

116 The authors acknowledge support from the US Air Force Office of Scientific Research through
 117 the Center of Excellence Grant FA9550-17-1-0195 (Technical Monitors: Fariba Fahroo, Mitat
 118 Birkan, Ramakanth Munipalli, Venkateswaran Sankaran).

119 We thank Nicholas Arnold-Medabalimi, Sahil Bhola, Cheng Huang, Ashish Nair, and Bernardo
 120 Pacini, and Elnaz Rezaian for their help in stress-testing PERFORM and providing valuable
 121 discussion on its development.

122 References

- 123 Carlberg, K., Barone, M., & Antil, H. (2017). Galerkin v. least-squares Petrov–Galerkin
 124 projection in nonlinear model reduction. *Journal of Computational Physics*, 330, 693–734.
 125 <https://doi.org/10.1016/j.jcp.2016.10.033>
- 126 Chollet, F., & others. (2015). *Keras*. <https://keras.io>.

- 127 Everson, R., & Sirovich, L. (1995). Karhunen–Loeve procedure for gappy data. *Journal of*
128 *the Optical Society of America A*, 12(8), 1657–1664. [https://doi.org/10.1364/JOSAA.](https://doi.org/10.1364/JOSAA.12.001657)
129 [12.001657](https://doi.org/10.1364/JOSAA.12.001657)
- 130 Grunloh, T., Patel, A., Lin, C., Wilson, T., Calian, L., Simonovic, P., & Safdari, M. (2021).
131 *AccelerateCFD Community Edition*. https://github.com/IllinoisRocstar/AccelerateCFD_CE.
- 132 Hess, M., & Rozza, G. (2020). *ITHACA-SEM - In real Time Highly Advanced Computational*
133 *Applications with Spectral Element Methods - Reduced Order Models for Nektar++*.
134 <https://github.com/mathLab/ITHACA-SEM>.
- 135 Hesthaven, J. S., Rozza, G., Stamm, B., & others. (2016). *Certified reduced basis methods*
136 *for parametrized partial differential equations* (Vol. 590). Springer. [https://doi.org/10.](https://doi.org/10.1007/978-3-319-22470-1)
137 [1007/978-3-319-22470-1](https://doi.org/10.1007/978-3-319-22470-1)
- 138 Huang, C., & Duraisamy, K. (2019). Investigation and improvement of robustness of reduced-
139 order models of reacting flow. *AIAA Scitech Forum*. <https://doi.org/10.2514/6.2019-2012>
- 140 Huang, C., Duraisamy, K., & Merkle, C. L. (2018). Challenges in reduced order modeling of re-
141 acting flows. *AIAA Propulsion and Energy Forum*. <https://doi.org/10.2514/6.2018-4675>
- 142 Huang, C., Wentland, C. R., Duraisamy, K., & Merkle, C. (2021). Model reduction for
143 multi-scale transport problems using model-form preserving least-squares projections with
144 variable transformation. *Journal of Computational Physics*, 448. [https://doi.org/10.1016/](https://doi.org/10.1016/j.jcp.2021.110742)
145 [j.jcp.2021.110742](https://doi.org/10.1016/j.jcp.2021.110742)
- 146 Lee, K., & Carlberg, K. T. (2020). Model reduction of dynamical systems on nonlinear
147 manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404.
148 <https://doi.org/10.1016/j.jcp.2019.108973>
- 149 Milk, R., Rave, S., & Schindler, F. (2016). pyMOR - generic algorithms and interfaces
150 for model order reduction. *SIAM Journal on Scientific Computing*, 38(5), S194–S216.
151 <https://doi.org/10.1137/15M1026614>
- 152 Rezaian, E., Huang, C., & Duraisamy, K. (2021). Non-intrusive balancing transformation of
153 highly stiff systems with lightly-damped impulse response. *arXiv:2109.10408*.
- 154 Rizzi, F., Blonigan, P. J., & Carlberg, K. T. (2020). Pressio: Enabling projection-based model
155 reduction for large-scale nonlinear dynamical systems. *arXiv:2003.07798*.
- 156 Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors, and dif-
157 ference schemes. *Journal of Computational Physics*, 43(2), 357–372. [https:](https://doi.org/10.1016/0021-9991(81)90128-5)
158 [//doi.org/10.1016/0021-9991\(81\)90128-5](https://doi.org/10.1016/0021-9991(81)90128-5)
- 159 Rowley, C. W., Colonius, T., & Murray, R. M. (2004). Model reduction for compressible flows
160 using POD and galerkin projection. *Physica D*, 189, 115–129. [https://doi.org/10.1016/](https://doi.org/10.1016/j.physd.2003.03.001)
161 [j.physd.2003.03.001](https://doi.org/10.1016/j.physd.2003.03.001)
- 162 Stabile, G., Hijazi, S., Mola, A., Lorenzi, S., & Rozza, G. (2017). POD-Galerkin reduced order
163 methods for CFD using finite volume discretisation: vortex shedding around a circular
164 cylinder. *Communications in Applied and Industrial Mathematics*, 8(1), 210–236. [https:](https://doi.org/10.1515/caim-2017-0011)
165 [//doi.org/10.1515/caim-2017-0011](https://doi.org/10.1515/caim-2017-0011)
- 166 Stabile, G., & Rozza, G. (2018). Finite volume POD-Galerkin stabilised reduced order methods
167 for the parametrised incompressible Navier-Stokes equations. *Computers & Fluids*. [https:](https://doi.org/10.1016/j.compfluid.2018.01.035)
168 [//doi.org/10.1016/j.compfluid.2018.01.035](https://doi.org/10.1016/j.compfluid.2018.01.035)
- 169 Venkateswaran, S., & Merkle, C. L. (1995). Dual time-stepping and preconditioning for
170 unsteady computations. *33rd Aerospace Sciences Meeting and Exhibit*. [https://doi.org/](https://doi.org/10.2514/6.1995-78)
171 [10.2514/6.1995-78](https://doi.org/10.2514/6.1995-78)