




# pyMARS: automatically reducing chemical kinetic models in Python

Phillip O. Mestas III<sup>1</sup>, Parker Clayton<sup>1</sup>, and Kyle E. Niemeyer<sup>1</sup>

<sup>1</sup> School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR USA 97331

DOI: [10.21105/joss.01543](https://doi.org/10.21105/joss.01543)

## Software

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

**Submitted:** 24 June 2019

**Published:** 08 September 2019

## License

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

## Summary

Chemically reacting fluid flows occur in a diverse range of scientific and engineering fields, including combustion and fire, atmospheric and oceanic fluid flows, electrochemical devices, heterogeneous catalysis, materials processing, and astrophysical fluid dynamics. Numerical simulations of reacting fluid flows rely on accurate chemical kinetic models to describe the participating chemical species and elementary reactions through which they interact. However, as models grow in detail they also grow in size, adding more species and reactions to capture intermediates and pathways. In the field of combustion, kinetic models for molecules relevant to transportation fuels (e.g., gasoline, diesel, jet fuel) can contain thousands of species and tens of thousands of elementary reactions (T. Lu & Law, 2009). Incorporating such models into multidimensional computational fluid dynamics simulations is practically impossible, due to the associated computational expense.

pyMARS, which stands for “Python-based Model Automatic Reduction Software”, is a software package that implements and applies literature methods for reducing chemical kinetic models, particularly targeted at combustion applications. pyMARS currently implements four “skeletal” reduction methods that identify and remove unimportant species and reactions: directed relation graph (DRG) (T. Lu & Law, 2005, 2006a, 2006b), directed relation graph with error propagation (DRGEP) (Niemeyer & Sung, 2011; Pepiot-Desjardins & Pitsch, 2008), path flux analysis (PFA) (Sun, Chen, Gou, & Ju, 2010), and sensitivity analysis (Niemeyer & Sung, 2015; Niemeyer, Sung, & Raju, 2010; Sankaran, Hawkes, Chen, Lu, & Law, 2007; Zheng, Lu, & Law, 2007). pyMARS succeeds the earlier Fortran-based MARS package (Niemeyer & Sung, 2014, 2015; Niemeyer et al., 2010), which used an in-house modified version of the proprietary Chemkin III library (Kee, Rupley, Meeks, & Miller, 1996).

## Background and features

DRG, DRGEP, and PFA represent the kinetic system as a graph, where nodes are species and directed, weighted edges represent the dependence of one species on another, through their participation in reactions. All three methods define interaction coefficients that approximate the error that would be induced in the overall production/consumption of one species if the other was removed from the model. To eliminate species, a cutoff threshold (e.g., 0.01–0.1) is applied to the system to eliminate unimportant connections or species. The methods differ in their definition of these coefficients and whether/how indirect relationships between species play a role. For all three methods, pyMARS iteratively increases the cutoff threshold until reaching a user-specified error limit.

Sensitivity analysis can be applied directly to a starting model, but generally it should be informed by and follow a graph-based method such as DRG or DRGEP (i.e., DRGASA and

DRGEPSA) due to the high cost associated with using such a brute-force approach on a large model. pyMARS implements two sensitivity analysis approaches: “initial” and “greedy” (Niemeyer & Sung, 2015). The initial algorithm finds the error induced by individually eliminating all species under consideration, one-by-one, then removes species in ascending order of induced error until reaching the limit. In contrast, the greedy algorithm reevaluates the induced error of remaining species at each step, ensuring that it uses the most-current information; this increases computational expense significantly, but generates a smaller reduced model.

Required inputs for pyMARS to perform model reduction include the starting chemical kinetic model in the standard Cantera (Goodwin, Speth, Moffat, & Weber, 2018) or Chemkin (Kee et al., 1996) formats (it first converts the latter to the former) and a YAML file with reduction parameters (including method and a maximum error limit to constrain the reduced model) and a list of initial conditions for homogeneous autoignition simulations. These simulations are used both to obtain ignition delay values for gauging the error of a candidate model, and also to sample thermochemical state data for the reduction methods (where 20 points are sampled during the ignition temperature rise). A graph-based reduction method can be specified using the `method` key, the `sensitivity-analysis` key can be specified as `True` to perform standalone sensitivity analysis, or both can be given to perform DRG/DRGEP/PFA-informed sensitivity analysis. Additional input keys include target species for DRG/DRGEP/PFA and optionally specifying a list of species to always retain.

Additional features include:

- Simulations can be parallelized via the `multiprocessing` module by adding the `--num_threads` command-line option and specifying a value greater than one. (Including the option alone will lead to pyMARS using the available number of threads minus one.)
- To save (potentially significant) time when performing multiple reductions for the same model, pyMARS saves and automatically reuses sampled autoignition data when possible.
- pyMARS provides a conversion functionality between the Cantera and Chemkin model formats, accessible by passing the `--convert` option.

pyMARS relies on the Cantera suite (Goodwin et al., 2018) to handle the chemical kinetics and perform autoignition simulations; it temporarily stores full simulation results as HDF5 files using PyTables (PyTables Developers Team, 2002–2019). It uses PyYAML (Simonov & others, 2018) to parse simulation input files, and NumPy (Walt, Colbert, & Varoquaux, 2011) arrays to store and manipulate data. Graph construction and searching rely on NetworkX (Hagberg, Schult, & Swart, 2008, 2019).

## Future work

Future work includes adding more reduction stages, including unimportant reaction elimination and incorporating the quasi-steady-state approximation for species (Niemeyer & Sung, 2015). In addition, we plan to add additional combustion phenomena for sampling and error evaluation, including one-dimensional laminar flame and perfectly stirred reactor simulations.

## Acknowledgements

This material is based upon work supported by the National Science Foundation under grant OAC-1535065.

## References

- Goodwin, D. G., Speth, R. L., Moffat, H. K., & Weber, B. W. (2018). Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes. <https://www.cantera.org>. doi:10.5281/zenodo.1174508
- Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference* (pp. 11–15). Pasadena, CA USA.
- Hagberg, A. A., Schult, D. A., & Swart, P. J. (2019). NetworkX. <https://github.com/networkx/networkx>.
- Kee, R. J., Rupley, F. M., Meeks, E., & Miller, J. A. (1996, May). CHEMKIN-III: A FORTRAN chemical kinetics package for the analysis of gas-phase chemical and plasma kinetics. Sandia National Laboratories Report SAND-96-8216. doi:10.2172/481621
- Lu, T., & Law, C. K. (2005). A directed relation graph method for mechanism reduction. *Proceedings of the Combustion Institute*, 30(1), 1333–1341. doi:10.1016/j.proci.2004.08.145
- Lu, T., & Law, C. K. (2006a). Linear time reduction of large kinetic mechanisms with directed relation graph: N-heptane and iso-octane. *Combustion and Flame*, 144(1-2), 24–36. doi:10.1016/j.combustflame.2005.02.015
- Lu, T., & Law, C. K. (2006b). On the applicability of directed relation graphs to the reduction of reaction mechanisms. *Combustion and Flame*, 146(3), 472–483. doi:10.1016/j.combustflame.2006.04.017
- Lu, T., & Law, C. K. (2009). Toward accommodating realistic fuel chemistry in large-scale computations. *Progress in Energy and Combustion Science*, 35(2), 192–215. doi:10.1016/j.pecs.2008.10.002
- Niemeyer, K. E., & Sung, C.-J. (2011). On the importance of graph search algorithms for DRGEP-based mechanism reduction methods. *Combustion and Flame*, 158(8), 1439–1443. doi:10.1016/j.combustflame.2010.12.010
- Niemeyer, K. E., & Sung, C.-J. (2014). Mechanism reduction for multicomponent surrogates: A case study using toluene reference fuels. *Combustion and Flame*, 161(11), 2752–2764. doi:10.1016/j.combustflame.2014.05.001
- Niemeyer, K. E., & Sung, C.-J. (2015). Reduced chemistry for a gasoline surrogate valid at engine-relevant conditions. *Energy & Fuels*, 29(2), 1172–1185. doi:10.1021/ef5022126
- Niemeyer, K. E., Sung, C.-J., & Raju, M. P. (2010). Skeletal mechanism generation for surrogate fuels using directed relation graph with error propagation and sensitivity analysis. *Combustion and Flame*, 157(9), 1760–1770. doi:10.1016/j.combustflame.2009.12.022
- Pepiot-Desjardins, P., & Pitsch, H. (2008). An efficient error-propagation-based reduction method for large chemical kinetic mechanisms. *Combustion and Flame*, 154(1-2), 67–81. doi:10.1016/j.combustflame.2007.10.020
- PyTables Developers Team. (2002–2019). PyTables: Hierarchical datasets in Python. Retrieved from <https://www.pytables.org/>
- Sankaran, R., Hawkes, E. R., Chen, J. H., Lu, T., & Law, C. K. (2007). Structure of a spatially developing turbulent lean methane–air Bunsen flame. *Proceedings of the Combustion Institute*, 31(1), 1291–1298. doi:10.1016/j.proci.2006.08.025
- Simonov, K., & others. (2018). PyYAML. <https://github.com/yaml/pyyaml/>.
- Sun, W., Chen, Z., Gou, X., & Ju, Y. (2010). A path flux analysis method for the reduction of detailed chemical kinetic mechanisms. *Combustion and Flame*, 157(7), 1298–1307. doi:10.1016/j.combustflame.2010.03.006

Walt, S. van der, Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22–30. doi:[10.1109/mcse.2011.37](https://doi.org/10.1109/mcse.2011.37)

Zheng, X. L., Lu, T. F., & Law, C. K. (2007). Experimental counterflow ignition temperatures and reaction mechanisms of 1,3-butadiene. *Proceedings of the Combustion Institute*, 31(1), 367–375. doi:[10.1016/j.proci.2006.07.182](https://doi.org/10.1016/j.proci.2006.07.182)