Arrhenius.jl: A Differentiable Combustion Simulation Package

Preprint · April 2021

DOI: 10.13140/RG.2.2.15356.87687

2 authors:

Weiqi Ji
Massachusetts Institute of Technology
32 PUBLICATIONS **304** CITATIONS
SEE PROFILE

Sili Deng
Massachusetts Institute of Technology
47 PUBLICATIONS **529** CITATIONS
SEE PROFILE

Arrhenius.jl: A Differentiable Combustion Simulation Package

Weiqi Ji[a], Sili Deng[a]

[a]*Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

Abstract

The uncertainties in the models and boundary conditions greatly influence the fidelity of combustion simulations, and therefore, uncertainty quantification has received much attention in recent years. Differentiable programming is a promising technique that can efficiently compute the gradients of loss functions with model parameters to evaluate the sensitivity in high-dimensional problems. However, it is often challenging to implement differentiable programming in practice, and thus it is still not available in widely utilized combustion simulation packages such as CHEMKIN and Cantera. Here, we present a differentiable combustion simulation package leveraging the differentiable programming eco-system in Julia, including DifferentialEquations.jl for solving differential equations, ForwardDiff.jl for forward-model auto-differentiation, Flux.jl for implementing neural network models into combustion simulations and optimizing neural network models using the state-of-the-art deep learning optimizers. We demonstrate the benefits of differential programming in efficient and accurate gradient computation, with applications in uncertainty quantification and model discovery.

Email addresses: weiqiji@mit.edu (Weiqi Ji), silideng@mit.edu (Sili Deng)

Keywords: HyChem, Chemical Reaction Neural Network, Machine Learning

1. Introduction

The optimization of model parameters is a crucial part of many tasks in numerical combustion simulations. While heuristic optimization methods, such as genetic algorithms, have been widely employed in combustion modeling, optimization algorithms based on stochastic gradient descent (SGD) have seldom been exploited. Meanwhile, SGD has shown promise in nonconvex optimization for complex nonlinear models, and SGD plays a central role in driving the bloom of deep learning in the last decade [1].

One of the major obstacles for exploiting SGD in combustion modeling is the lack of software ecosystems that can efficiently and accurately compute the gradient of simulation output to model parameters. For instance, the finite difference method (often termed as the brute-force method) usually suffers both computational inefficiency for the cost scales with the number of parameters, and inaccuracy due to the truncation error. Conversely, gradient evaluation methods based on auto-differentiation (AD) have shown both efficiency and accuracy in the training of large-scale deep neural network models. Many open-source AD packages have been developed in the last decade, to name a few, TensorFlow and Jax backed by Google, PyTorch backed by Facebook, ForwardDiff.jl [2] and Zygote.jl [3] in Julia. To this end, we introduce the AD-powered differentiable combustion simulation package Arrhenius.jl [4], which incorporates the combustion physics models into AD ecosystems in Julia to facilitate the research of differentiable combustion modeling.

2. Arrhenius.jl

Arrhenius.jl is built using the programming language of Julia to leverage the rich ecosystems of auto-differentiation and differential equation solvers. Arrhenius.jl does two types of differentiable programming: (i) it can differentiate elemental computational blocks. For example, it can differentiate the reaction source term with respect to kinetic and thermodynamic parameters as well as species concentrations. (ii) It can differentiate the entire simulator in various ways, such as solving the continuous sensitivity equations [5] as did in CHEMKIN [6] and Cantera [7] and in adjoint methods [8, 9]. The first type of differentiation is usually the basis of the second type of high-level differentiation. Arrhenius.jl offers the core functionality of combustion simulations in native Julia programming, such that users can conveniently build the applications on top of Arrhenius.jl and exploit various approaches to do high-level differentiation.

Figure 1 shows a schematic diagram of the structure of the Arrhenius.jl package. The Arrhenius.jl reads in the chemical mechanism files in YAML format maintained by the Cantera community, and the chemical mechanism files contain the kinetic models, thermodynamic, and transport databases. The core functionality of Arrhenius.jl is to compute the reaction source terms and mixture properties, such as heat capacities, enthalpies, entropies, Gibbs free energies, etc. In addition, Arrhenius.jl offers flexible interfaces for users to define neural network models as submodels and augment them with existing physical models. For example, one can use a neural network submodel to represent the unknown reaction pathways and exploit various scientific machine learning methods to train the neural network models, such as neural

ordinary differential equations [10, 11] and physics-informed neural network models [12, 13]. One can then implement the governing equations for different applications with these core functionalities and solve the governing equations using classical numerical methods or neural network-based solvers, such as physics-informed neural networks [12]. Arrhenius.jl provides solvers for canonical combustion problems, such as simulating the auto-ignition in constant volume/pressure reactors and oxidation in jet stirred reactors.

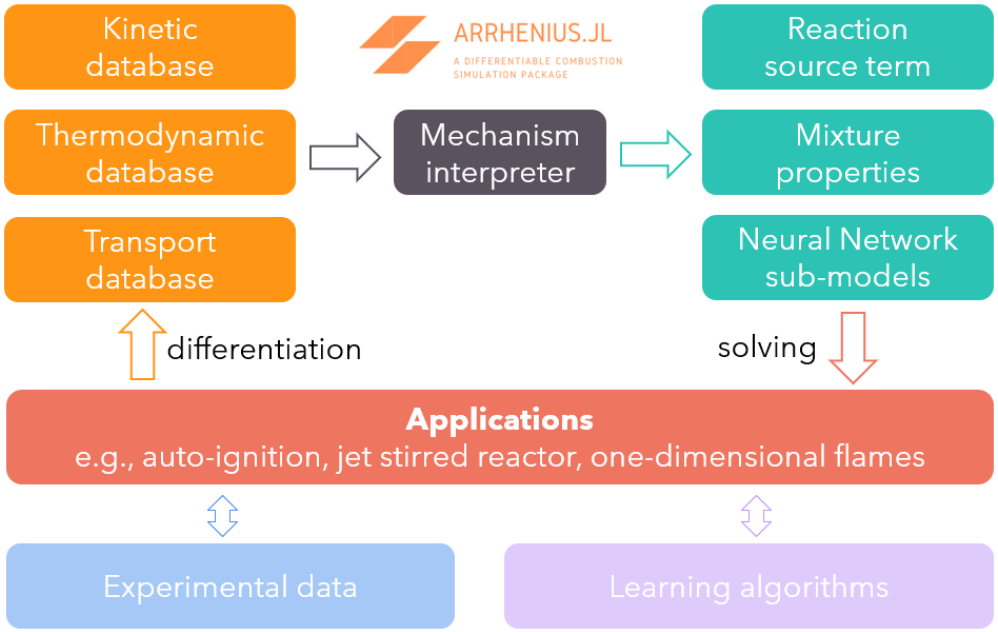

Figure 1: Schematic diagram showing the structure of the Arrhenius.jl package.

Compared to the legacy combustion simulation packages, Arrhenius.jl can not only provide predictions given the physical models, but also optimize model parameters given experimental measurements. By efficiently and accurately evaluating the gradient of the solution outputs to the model pa-

rameters, experimental data can be incorporated into the simulation pipeline to enable data-driven modeling with deep learning algorithms.

3. Results and Discussion

We then present three case studies of Arrhenius.jl in uncertainty quantification and model discovery.

3.1. Application in Uncertainty Quantification

We then apply Arrhenius.jl in computing the active subspace for quantifying kinetic uncertainties. Active subspace [14] is a parametric dimension reduction approach that identifies the low-dimensional subspace of high-dimensional uncertain parameters via the singular value decomposition of the parameter's gradient space. It has been applied to evaluate combustion model uncertainties in zero/one-dimensional simulations [15] as well as in turbulent combustion simulations [16]. We used the chemical models of GRI3.0 and LLNL detailed n-heptane mechanism (Version 3.1) [17] for demonstrations to compute eigenvalue spectra. In Figs. 2a and c, one-dimensional active subspace is identified for both cases, since the first eigenvalue is larger than the second by two orders of magnitude. The summary plots in Figs. 2b and d further show that the variations of the IDT can be well captured by the first active direction, defined by the leading eigenvector. The active variable refers to the projected value of the samples on the first active direction. The gradients of IDT are computed via sensBVP method [18]. As far as the computational cost is concerned, our approach obtained the active subspace within one minute on a laptop for GRI3.0, while the finite difference approach takes about an hour. With the identified one-dimensional

active subspace, one can exploit it for global sensitivity analysis, forward and inverse uncertainty quantification, and optimization.

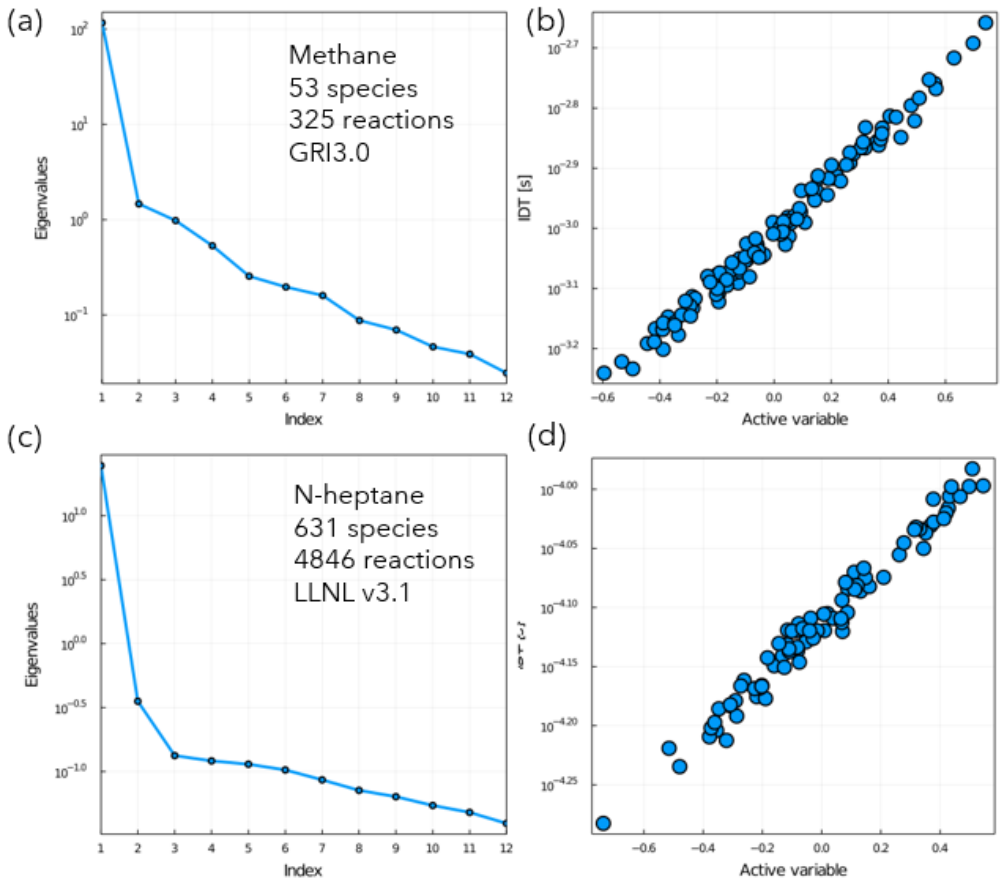

Figure 2: (a, c) Eigenvalue spectra and (b, d) summary plots for the mechanism of GRI3.0 and LLNL v3.1 for n-heptane. The thermodynamic conditions for both cases are under 40 atm, 1200 K, and stoichiometric fuel/air mixture. The uncertainty considered are the uncertainties in the pre-factor A. An independent log-uniform distribution is assumed for all of the reactions, and $ln(A/A_0) \sim \mathbf{U}[-0.5, 0.5]$. The number of samples drawn from the parameter space is set as $20 * log(nr)$ for methane, $10 * log(nr)$ for n-heptane, and nr is the number of reactions.

3.2. Application in Scientific Machine Learning

Finally, we present the application in Scientific Machine Learning (SciML) [11] by using Arrhenius.jl to develop a neural-network-based pyrolysis submodel within the HyChem model framework [19]. Our recently developed Chemical Reaction Neural Network (CRNN) [20] approach is employed to develop the neural network model for its interpretability, such that the learned model complies with fundamental physical laws and provides insights, as well as its compatibility with CHEMKIN/Cantera packages. The conventional HyChem-based pyrolysis submodels require expert knowledge of the chemical kinetics which takes years to develop. On the contrary, the CRNN approach aims at autonomously discover the reaction pathways and kinetic parameters simultaneously to accelerate high-fidelity chemical model development. In the following demonstration, the CRNN-HyChem approach is utilized to model the jet fuel of JP10.

As shown in Fig. 3, the CRNN-HyChem approach models the fuel chemistry of JP10 with two submodels, similar to the convention HyChem concept. The CRNN submodel models the breakdown of JP10 fuel molecules into smaller hydrocarbons up to C_6H_6 and the submodel for C_0 - C_6 describes the oxidation chemistry. For proof-of-concept, we chose the same species in the original HyChem pyrolysis submodel [19] to be included in the CRNN model; however, it should be noted that the such choice of species can be treated as hyper-parameters to circumvent the need for expert knowledge and achieve potentially better performance. In the original CRNN approach [20], the Law of Mass Action and Arrhenius Law are enforced by the design of the structure of the neural network. Reaction orders are assumed to be equal

to the stoichiometric coefficients for the reactants. In the present study, elemental conservation is further guaranteed by projecting the stoichiometric coefficients into the elemental conservation space. For better convergence, the stoichiometric coefficients for JP10 is fixed as -1, and during the training, the stoichiometric coefficients are regularized to achieve better numerical stability. The training data were generated by simulating the IDT using the original JP10 HyChem model. A wide range of thermodynamic conditions were considered: the pressures of 1-60 atm, the initial temperatures of 1100-1800 K, and the equivalence ratios of 0.5-1.5. In total, 500 thermodynamic conditions were randomly generated using the Latin hypercube sampling method. The dataset was split into training and validation datasets with a ratio of 70:30.

The learned stoichiometric coefficients and kinetic parameters are shown in Table 1, where negative and positive stoichiometric coefficients correspond to reactants and products, respectively.Qualitatively, most of the learned pathways are H-abstract reactions which is consistent with the expert-derived HyChem models. However, quantitatively the learned pathways are not the same as HyChem model, and further efforts will be directed to extracting physical insights from the learned pathways. Figure 4 then compares the results of the learned CRNN model and the label data generated by using the original HyChem model [19], and they agree very well. The results thus demonstrate the capability of Arrhenius.jl in learning augmented neural network models with hundreds of parameters, which is impossible with finite difference methods.

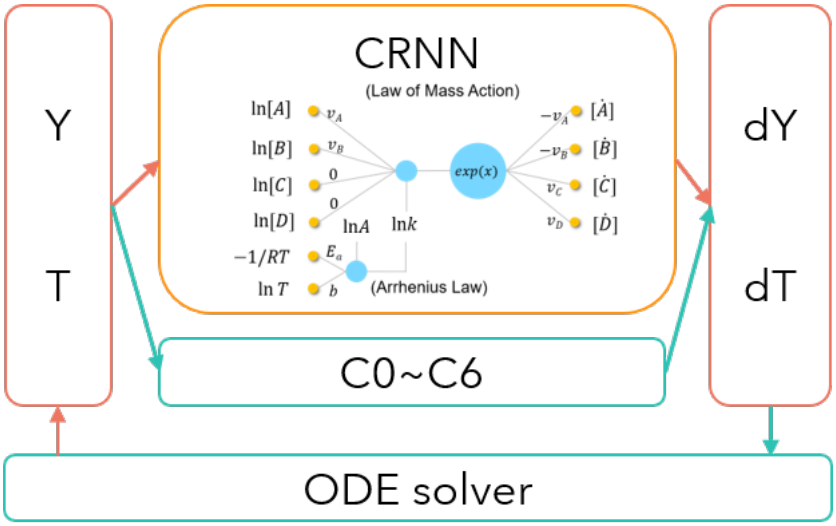

Figure 3: Schematic diagram showing the structure of the CRNN-HyChem approach.

Table 1: Learned stoichiometric coefficients and kinetic parameters. Reaction orders are assumed to be equal to the stoichiometric coefficients for reactants.

Stoichiometric Coefficients														Ea	b	ln (A)
C10H16	H	CH3	OH	HO2	O	CH4	C2H4	C3H6	C5H6	C6H6	H2O2	O2	H2O	(cal/mol)		
-1	0.11	0.32	0.06	-0.08	-0.12	0.44	0.42	0.63	0.6	0.59	0.12	-0.11	0.18	24.52	-0.29	15.33
-1	0.12	0.33	0.1	0.08	-0.16	0.46	0.43	0.65	0.6	0.57	-0.09	-0.07	0.23	19.93	-0.1	16.06
-1	0.33	0.26	-0.09	-0.02	-0.18	0.5	0.45	0.7	0.6	0.54	-0.12	0.16	0.22	17.63	0.07	18.48
-1	0.1	0.31	-0.09	0.03	-0.24	0.44	0.42	0.63	0.61	0.58	0.14	-0.11	0.2	19.04	-0.26	17.21
-1	0.14	0.3	0.1	-0.14	-0.23	0.44	0.44	0.58	0.6	0.61	0.16	-0.07	0.23	17.09	-0.05	17.2
-1	0.35	0	0	-0.17	0.01	0.55	0.77	0.65	0.37	0.68	-0.07	0.12	0.23	14.45	0.12	18.64
-1	0.14	0.34	-0.02	0.06	-0.15	0.46	0.43	0.66	0.6	0.56	-0.07	-0.02	0.23	19.10	-0.19	16.3
-1	0.2	0.21	0.08	-0.18	0.05	0.53	0.54	0.64	0.5	0.63	-0.05	0.04	0.26	16.41	-0.08	17.37
-1	0.1	0.3	-0.16	0.04	-0.29	0.44	0.41	0.63	0.61	0.58	0.17	-0.09	0.22	21.74	-0.1	17.72
-1	0.52	0.28	0.01	0	0	0.76	0.39	0.29	0.98	0.4	-0.23	0.11	0.23	12.98	0	20.09

4. Conclusions

This work presents a differential combustion simulation package called Arrhenius.jl, which can perform efficient and accurate gradient evaluations across classical reacting flow solvers. We expect that the open-source package could greatly facilitate the integration of modern machine learning techniques into combustion modeling, especially the scientific machine learning that

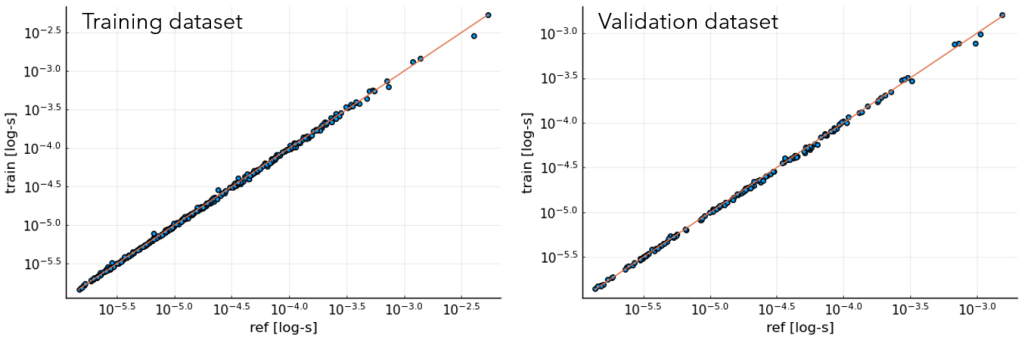

Figure 4: Comparisons between the predicted ignition delay times using the learned CRNN model and the HyChem model for both the training and validation datasets.

takes the advantage of both physical and data-driven modeling. We also invite the contribution from the reacting flow community to enhance the capability of the package and explore its potential in other applications.

References

[1] Y. Bengio, I. Goodfellow, A. Courville, Deep learning, Vol. 1, MIT press Massachusetts, USA:, 2017.

[2] J. Revels, M. Lubin, T. Papamarkou, Forward-mode automatic differentiation in julia, arXiv preprint arXiv:1607.07892 (2016).

[3] M. Innes, Don't unroll adjoint: Differentiating ssa-form programs, arXiv preprint arXiv:1810.07951 (2018).

[4] W. Ji, S. Deng, Arrhenius.jl: A differentiable combustion simulation package, `https://github.com/DENG-MIT/Arrhenius.jl` (2021).

[5] W. Ji, Z. Ren, C. K. Law, Evolution of sensitivity directions during

autoignition, Proceedings of the Combustion Institute 37 (1) (2019) 807–815.

[6] R. J. Kee, F. M. Rupley, J. A. Miller, Chemkin-ii: A fortran chemical kinetics package for the analysis of gas-phase chemical kinetics, Tech. rep., Sandia National Lab.(SNL-CA), Livermore, CA (United States) (1989).

[7] D. G. Goodwin, H. K. Moffat, R. L. Speth, Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes (2009).

[8] C. Rackauckas, Y. Ma, V. Dixit, X. Guo, M. Innes, J. Revels, J. Nyberg, V. Ivaturi, A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions, arXiv preprint arXiv:1812.01892 (2018).

[9] C. Rackauckas, Q. Nie, Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia, Journal of Open Research Software 5 (1) (2017).

[10] R. T. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, arXiv preprint arXiv:1806.07366 (2018).

[11] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, A. Edelman, Universal differential equations for scientific machine learning, arXiv preprint arXiv:2001.04385 (2020).

[12] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse

problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[13] W. Ji, W. Qiu, Z. Shi, S. Pan, S. Deng, Stiff-pinn: Physics-informed neural network for stiff chemical kinetics, arXiv preprint arXiv:2011.04520 (2020).

[14] P. G. Constantine, E. Dow, Q. Wang, Active subspace methods in theory and practice: applications to kriging surfaces, SIAM Journal on Scientific Computing 36 (4) (2014) A1500–A1524.

[15] W. Ji, J. Wang, O. Zahm, Y. M. Marzouk, B. Yang, Z. Ren, C. K. Law, Shared low-dimensional subspaces for propagating kinetic uncertainty to multiple outputs, Combustion and Flame 190 (2018) 146–157.

[16] W. Ji, Z. Ren, Y. Marzouk, C. K. Law, Quantifying kinetic uncertainty in turbulent combustion simulations using active subspaces, Proceedings of the Combustion Institute 37 (2) (2019) 2175–2182.

[17] M. Mehl, W. J. Pitz, C. K. Westbrook, H. J. Curran, Kinetic modeling of gasoline surrogate components and mixtures under engine conditions, Proceedings of the Combustion Institute 33 (1) (2011) 193–200.

[18] V. Gururajan, F. N. Egolfopoulos, Direct sensitivity analysis for ignition delay times, Combustion and Flame 209 (2019) 478–480.

[19] Y. Tao, R. Xu, K. Wang, J. Shao, S. E. Johnson, A. Movaghar, X. Han, J.-W. Park, T. Lu, K. Brezinsky, et al., A physics-based approach to modeling real-fuel combustion chemistry–iii. reaction kinetic model of jp10, Combustion and Flame 198 (2018) 466–476.

[20] W. Ji, S. Deng, Autonomous discovery of unknown reaction pathways from data by chemical reaction neural network, The Journal of Physical Chemistry A 125 (4) (2021) 1082–1092.