pyJac: Analytical Jacobian generator for chemical kinetics

Kyle E. Niemeyer a,*, Nicholas J. Curtis b, Chih-Jen Sung b

a School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR 97331, USA
b Department of Mechanical Engineering, University of Connecticut, Storrs, CT, 06269, USA

Abstract

Accurate simulations of combustion phenomena require the use of detailed chemical kinetics in order to capture limit phenomena such as ignition and extinction as well as predict pollutant formation. However, the chemical kinetic models for hydrocarbon fuels of practical interest typically have large numbers of species and reactions and exhibit high levels of mathematical stiffness in the governing differential equations, particularly for larger fuel molecules. In order to integrate the stiff equations governing chemical kinetics, generally reactive-flow simulations rely on implicit algorithms that require frequent Jacobian matrix evaluations. Some in situ and posteriori computational diagnostics methods also require accurate Jacobian matrices, including computational singular perturbation and chemical explosive mode analysis. Typically, finite differences numerically approximate these, but for larger chemical kinetic models this poses significant computational demands since the number of chemical source term evaluations scales with the square of species count. Furthermore, existing analytical Jacobian tools do not optimize evaluations or support emerging SIMD processors such as GPUs. Here we introduce pyJac, a Python-based open-source program that generates analytical Jacobian matrices for use in chemical kinetics modeling and analysis. In addition to producing the necessary customized source code for evaluating reaction rates (including all modern reaction rate formulations), the chemical source terms, and the Jacobian matrix, pyJac uses an optimized evaluation order to minimize computational and memory operations. As a demonstration, we first establish the correctness of the Jacobian matrices for kinetic models of hydrogen, methane, ethylene, and isopentanol oxidation (number of species ranging 13–360) by showing agreement within 0.001% of matrices obtained via automatic differentiation. We then demonstrate the performance achievable on CPUs and GPUs using pyJac via matrix evaluation timing comparisons; the routines produced by pyJac outperformed first-order finite differences by 3–7.5 times and the existing analytical Jacobian software TChem by 1.1–2.2 times on a single-threaded basis. It is noted that TChem is not thread-safe, while pyJac is easily parallelized, and hence can greatly outperform TChem on multicore CPUs. The Jacobian matrix generator we describe here will be useful for reducing the cost of integrating chemical source terms with implicit algorithms in particular and algorithms that require an accurate Jacobian matrix in general. Furthermore, the open-source release of the program and Python-based implementation will enable wide adoption.

Program summary

Program Title: pyJac
Program Files doi: http://dx.doi.org/10.17632/mm3z8j2m5.1
Licensing provisions: MIT License
Programming language: Python
External routines/libraries: required: NumPy; optional: Cython, Cantera, PyYAML, Adept
Nature of problem: Automatic generation of source code to evaluate Jacobian matrices for chemical kinetic models

* This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail address: Kyle.Niemeyer@oregonstate.edu (K.E. Niemeyer).
http://dx.doi.org/10.1016/j.cpc.2017.02.004
0010–4655/© 2017 Elsevier B.V. All rights reserved.

Please cite this article in press as: K.E. Niemeyer, et al., pyJac: Analytical Jacobian generator for chemical kinetics, Computer Physics Communications (2017), http://dx.doi.org/10.1016/j.cpc.2017.02.004
1. Introduction

As the need for detailed and accurate chemical kinetic models in predictive reactive-flow simulations has become recognized in recent years, such models describing the oxidation of hydrocarbon fuels simultaneously grew orders of magnitude in size and complexity. For example, a recently developed detailed kinetic model for 2-methylalkanes, relevant for jet and diesel fuel surrogates, consists of over 7000 species and 30,000 reactions [1]; similarly large surrogate models exist for gasoline [2,3] and biodiesel [4]. Since in general the computational cost of solving the associated generally stiff systems of equations scales quadratically with the number of species at best—and at worst, cubically—models of such a large size pose challenges even for lower-dimensional analyses, and cannot practically be used directly in multidimensional reactive-flow simulations.

In an effort to reduce the computational demands of using large, detailed kinetic models, a number of techniques have been developed to reduce their size and complexity while retaining predictiveness, as reviewed by Lu and Law [5] as well as Turányi and Tomlin [6]. Major classes of such approaches include skeletal reduction methods that remove unimportant species and reactions [7–10], lumping of species that share similar properties [11–13], and time-scale reduction methods that reduce chemical stiffness [14–17]. Effective reduction strategies either combine multiple methods a priori [18–20] or apply them dynamically during a simulation to achieve greater local savings [21–26]. Techniques such as interpolation/tabulation of expensive terms [27] can also reduce computational costs.

In addition to the aforementioned cost reduction methods that modify the chemical kinetic models, improvements to the integration algorithms that actually solve the governing differential equations can also offer significant gains in computational performance. Due to the chemical stiffness exhibited by most kinetic models, solvers typically rely on robust, high-order implicit integration algorithms based on backward differentiation formulas [28–31]. In order to solve the nonlinear algebraic equations that arise in these methods, the Jacobian matrix must be evaluated and factorized, operations that result in the quadratic and cubic costs mentioned previously. However, by using an analytical formulation for the Jacobian matrix rather than a typical finite difference approximation, the cost of the numerous evaluations can drop from growing with the square of the number of species to a linear dependence [5].

In parallel with the potential improvements in stiff implicit integrators used for chemical kinetics, algorithms tailored for high-performance hardware accelerators offer another route to reducing computational costs. In the past, central processing unit (CPU) clock speeds increased regularly—i.e., Moore’s Law—but power consumption and heat dissipation issues disrupted this trend recently, slowing the pace of increases in CPU clock rates. While multicore parallelism continues to raise CPU performance, single-instruction multiple data (SIMD) processors, e.g., graphics processing units (GPUs), recently emerged as a low-cost, low-power consumption, and massively parallel high-performance computing alternative. GPUs—originally developed for graphics/video processing and display purposes—consist of hundreds to thousands of cores, compared to the tens of cores found on a typical CPU. Recognizing that the SIMD parallelism model fits well with the operator-split chemistry integration that forms the basis of many reactive-flow codes [32], a number of studies in recent years [33–38] explored the use of SIMD processors to accelerate the integration of chemical kinetics in reactive-flow codes. Niemeyer and Sung [39] reviewed such efforts in greater detail. While explicit methods offer significant improvements in performance for non-stiff and moderately stiff chemical kinetics [38], experiences thus far suggest that stiff chemistry continues to require the use of implicit or similar algorithms. This provides significant motivation to provide the capability of evaluating analytical Jacobian matrices on GPUs as well as CPUs.

Thus, motivated by the potential cost reductions offered by analytical Jacobian matrix formulations, over the past five years a number of research groups developed analytical Jacobian generators for chemical kinetics, although as will be discussed the software package introduced here offers a number of improvements. The TChem toolkit developed by Safta et al. [40] was one of the first software packages developed for calculating the analytical Jacobian matrix, but provides this functionality through an interface rather than generating customized source code for each model. Youseffi [41] recognized the importance of using an analytical Jacobian over a numerical approximation to reduce both computational cost and numerical error when performing eigen-decomposition of the matrix. Bisetti [42] released a utility for producing analytical Jacobian matrix source code based on isothermal and isobaric conditions, with the state vector comprised of species concentrations rather than mass fractions; while incompatible with many existing reactive-flow formulations, this formulation resulted in a significant increase of Jacobian matrix sparsity—such a strategy should be investigated further. Perini et al. [43] developed an analytical Jacobian matrix formulation for constant-volume combustion, and when used in a multi-dimensional reactive-flow simulation—combined with tabulation of temperature-dependent properties—reported a performance improvement of around 80% over finite-difference-based approximations. Recently, Dijkmans et al. [44] used a GPU-based analytical Jacobian combined with tabulation of temperature-dependent functions based on polynomial interpolations to accelerate the integration of chemical kinetics equations, similar to the earlier approach of Shi et al. [34]. Unlike the current work, the approach of Dijkmans et al. [44] used the GPU to calculate the elements of a single Jacobian matrix in parallel, rather than a large number of matrices corresponding to different states.

To our knowledge, currently no open-source analytical chemical Jacobian tool exists that is capable of generating code specifically optimized for SIMD processors. To this end, pyJac is capable of generating subroutines for reaction rates, species production rates, derivative source terms, and analytical

---

1 Note that the term “reaction mechanism” is also used commonly in the literature; we adopted our preferred terminology “chemical kinetic model” here.
دریافت فوری متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات