Sub Topic: Reaction Kinetics

10th U.S. National Combustion Meeting
Organized by the Eastern States Section of the Combustion Institute
April 23–26, 2017
College Park, Maryland

Investigating stiffness detection metrics for chemical kinetics

Andrew Alferman[1] *and Kyle E. Niemeyer*[1,*]

[1] *School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR 97331, USA*
[*] *Corresponding author:* Kyle.Niemeyer@oregonstate.edu

Abstract: Simulations of combustion and reactive flows often exhibit numerical stiffness in the equations governing chemical kinetics. Explicit solvers for these equations offer low computational expense, but typically cannot handle stiff systems of differential equations. In contrast, implicit methods demand greater expense but offer unconditional stability—as a result, most non-DNS reactive flow solvers rely on these methods by default. However, explicit or stabilized explicit methods can instead be used to reduce the computational expense while remaining stable and accurate if the chemical kinetics systems exhibit low-to-moderate stiffness. This study aims to investigate stiffness quantification metrics, with the goal of identifying one capable of efficiently and robustly determining the appropriate category of integrator required. Literature on stiffness quantification will be surveyed and methods of measuring the stiffness of chemical kinetics states will be investigated, including methods based on eigendecomposition or the spectral radius of the Jacobian matrix, error estimations, conditioning parameters, and computational cost estimations. These methods will be applied during the solution of hydrogen and carbon monoxide autoignition with different initial conditions based on partially stirred reactor simulation sampling, and evaluated in terms of effectiveness and computational efficiency.

Keywords: Stiffness quantification, Ordinary differential equations, Chemical kinetics, Computational cost reduction

1. Introduction

An increasing demand for greater efficiency and lower CO_2 emissions in the US energy supply has driven the development of next-generation combustion technologies that use alternative fuels [1]. Computational modeling is one important tool that increasingly drives design and development of these new technologies. Shortening the design cycle and time-to-market of novel and efficient energy devices optimized for alternative fuels requires fundamental advances in combustion science and fuel chemistry; both depend on high-fidelity computational tools [2]. The need to develop new, high-performance computational methods to support combustion research has been recognized by multiple federal agencies as a key objective towards implementing predictive models [3–6].

Species time scales can range from the order of nanoseconds to seconds, requiring greater computational expense for integration by conventional methods (i.e., numerical stiffness) [7]. Most combustion modeling approaches rely on a single, implicit ODE integration method to handle the chemistry in all spatial locations, but these methods are computationally expensive, especially for larger mechanisms. The computational cost of chemistry is either a quadratic or cubic function of the number of species [7]; reaction mechanisms of sizes comparable to those shown above pose computational difficulties even in zero-dimensional (homogeneous) simulations.

Depending on the local conditions, computational fluid dynamics flow time-step size, and chemical mechanism being used, it is possible to encounter a wide range of stiffness within a single simulation. We can exploit this situation to reduce the overall simulation expense by selecting the most computationally efficient ODE solver on-the-fly based on local conditions. For example, a low-cost explicit method may be used far away from the reaction zone, while an implicit—or otherwise "stiff"—solver is more economical inside a flame. Making this selection requires a method to detect and classify stiffness [2].

Although the concept of stiffness has been identified for over 60 years, the term has not been precisely defined despite repeated efforts. The diverse set of problems considered to be "stiff" and the large variety of characterizations used to describe stiffness are amongst the difficulties that have been encountered in developing a precise definition [8]. Nonetheless, a variety of stiffness quantification methods have been developed with the goal of providing a practical means of evaluating a systems ODEs [8–12]. We look to these methods of stiffness quantification to determine their usefulness with respect to the equations governing combustion, and to determine if a reliable and efficient means of switching methods can be developed from it. In doing so, we hope to reduce the computational expense of combustion simulations.

One method of interest regarding stiffness quantification was the "IA-Stiffness Index" proposed by Shampine [13]. This method enables comparison of values of the stiffness index between different equations and different methods. Additionally, the method takes into account the impact of the order of the method selected when evaluating the stiffness. The method is relatively straightforward to implement, requiring computation of a vector of the derivatives of the system of equations, as well as either a weighted norm or the spectral radius of the Jacobian matrix.

2. Methods

We selected the relatively small hydrogen/carbon monoxide (H_2/CO) model of Burke et al. [14] for this initial investigation into stiffness quantification. This system only involves 13 chemical species, and therefore the Jacobian matrix need only be 14×14 after the temperature derivatives are included. This small Jacobian matrix allows for much faster evaluations of stiffness than would be possible with larger, more complicated kinetic systems, which can have hundreds to hundreds of thousands of chemical species [15]. The stiffness quantification methods investigated using the H_2/CO system can easily be applied to more complicated models.

Our analysis was conducted using information gathered from partially stirred reactor (PaSR) simulations. As described by Niemeyer et al. [16], the PaSR model consists of a number of particles, each with a time-varying composition. At discrete time steps, events including inflow, outflow, and pairing cause particles to change composition; between these time steps, mixing and reaction fractional steps evolve the composition of all particles. By using data from a single particle at a single time step of the PaSR model, the evaluation could be simplified to a zero-dimensional analysis. The stiffness detection metrics and method-switching algorithm developed in the zero-dimensional analysis may be applied to more complicated models in future work. (Please refer to Niemeyer et al. [16] for a detailed description of the relevant equations and solution technique.)

The IA-stiffness index of a method of order p introduced by Shampine [13] is

$$\frac{h_{acc}}{h_{iter}} \doteq \tau^{1/(p+1)} \rho[f_y(x_n, y(x_n))] \| y^{(p+1)}(x_n) \|^{-1/(p+1)} \left(\frac{|\xi|^{-1/(p+1)}}{|\gamma|} \right) \tag{1}$$

or alternatively,

$$\frac{h_{acc}}{h_{iter}} \doteq \tau^{1/(p+1)} \|f_y(x_n,y(x_n))\| \|y^{(p+1)}(x_n)\|^{-1/(p+1)} \left(\frac{|\xi|^{-1/(p+1)}}{|\gamma|} \right) \tag{2}$$

where h_{acc} represents the largest step size which would result in a local accuracy test being passed, h_{iter} represents the minimum step size that will lead to divergence of simple iteration, τ represents the specified tolerance, $\rho[f_y(x_n,y(x_n))]$ represents the spectral radius of the Jacobian matrix $f_y(x_n,y(x_n))$, γ is a constant characteristic of the formula, and ξ represents a constant characterizing the accuracy of a reference method. Note that these equations require the $p+1$ derivative of the function $y(x_n)$. Additionally, the matrix norm $\|M\|$ of matrix M with i columns and j rows is given by

$$\|M\| = \max_i \frac{1}{w_i} \sum_j |M_{ij}| w_j \tag{3}$$

in which w_i and w_j are positive weights of the matrix [9].

Although the above equations are useful in determining the stiffness of a method for a given system of equations, we are interested in quantifying the stiffness inherent to the system of equations itself. Such a quantification will become necessary in further research when method switching mechanisms are investigated. Shampine notes that the scalar quantity

$$\text{index} = \rho[f_y(x_n,y(x_n))] \|y^{(p+1)}(x_n)\|^{-1/(p+1)} \tag{4}$$

provides a fair “stiffness index” to a given problem [9]. For the remainder of this document, all references to the stiffness index will refer to the value obtained using equation (4).

As previously noted, use of the above equations requires calculation of a vector of the derivatives of a system of equations. In the case of this investigation, this vector is comprised of the derivatives of the thermochemical composition vector with respect to time, with the the vector defined as

$$\Phi = \{T, Y_1, Y_2, \ldots, Y_{N_{sp}}\}^{\mathsf{T}} , \tag{5}$$

where T is the temperature, Y_i are the species mass fractions, and N_{sp} is the number of species in the mechanism [16]. The vector needed to compute the stiffness index is therefore

$$\frac{\partial \Phi}{\partial t} = \left\{ \frac{\partial Y_1}{\partial t}, \frac{\partial Y_2}{\partial t}, \ldots, \frac{\partial Y_{N_{sp}}}{\partial t} \right\}^{\mathsf{T}} , \tag{6}$$

where time is denoted by t [16]. In addition to this vector of derivatives, computation of the stiffness index requires either the weighted norm or the spectral radius of the Jacobian matrix for the thermochemical composition vector. Computation of the vector of derivatives and the Jacobian matrix was readily achieved using pyJac [17], a Python-based open-source program that generates analytical Jacobian matrices for use in chemical kinetics modeling and analysis [16]. pyJac uses a chemical mechanism developed using Cantera [18] software and a set of initial conditions to generate functions that return the required values. After generating this data using pyJac, the values of the second derivative of the thermochemical composition vector were calculated numerically using a fourth-order central differencing formula. Higher-level derivatives may also be generated using the same numerical approach. Use of the central differencing formula was possible because the stiffness index was calculated after solving for the thermochemical composition vector, and the reaction was modeled using time steps of a constant size. Forward and backward difference formulae were used at the boundaries where central differencing could not be used. Variable time step methods for calculating the second derivative or higher-level derivatives may also be used, however these methods were not necessary at the current stage of the investigation.

The code used to calculate the stiffness index was written in Python and made use of open-source software including the `odeint` integrator function of SciPy [19]. This integrator uses the LSODA solver from the FORTRAN library `odepack` [20], which incidentally has its own algorithm for automatically switching between a non-stiff and a stiff solver. To ensure a high degree of accuracy, the integrator was given values of 10^{-13} and 10^{-15} for the error control parameters `relerr` and `abserr`, respectively. `NumPy` [21] was also extensively used in the code to calculate the stiffness index. At the current stage of this investigation, the selection of the integrator has no impact on the evaluation of the stiffness index due to the a posteriori approach. Rather than determining the higher-level derivatives based on the order of the solver used, the order of the method is manually entered into the code. To facilitate comparison of the results obtained by Shampine [9], the order of the method was assumed to be 1. To ensure that the code was running properly, the van der Pol equation with a ε value of 1000 was used as a test equation.

3. Results and Discussion

The initial condition of the evaluation was taken from a particle and time in the PaSR simulation, where weak reactions were taking place but ignition had not yet occurred. A 0.5 second simulation was run by integrating the derivative vector function. Data obtained from the integration demonstrates that autoignition was achieved with a residence time of approximately 0.11 seconds. The initial temperature of the simulation was approximately 865 K, which rapidly increases to 2440 K within 0.001 s (i.e., autoignition). After ignition, the temperature remains nearly constant for the remainder of the simulation. The mass fractions of all of the species considered in the model exhibited similar trends; each rapidly changing then becoming nearly constant immediately after ignition. The data obtained from the integration was then used to compute the stiffness index for every time step of the integration.

To minimize the computational expense of determining the stiffness index of a given mechanism, a solver can be modified to compute the index at the same time that it calculates values of the solution. The algorithm would then maximize the use of values already computed to provide the solution value, which makes it most desirable for use with algorithms making use of Jacobian matrices and values of the time derivatives.

3.1 Results

Values of the stiffness index varied between 10^{-1} and 10^{10} throughout the evaluation. The stiffness index of the reaction was greatest a short time after the reaction had stabilized following ignition, with values on the order of 10^{9}–10^{10}. For comparison, the maximum stiffness index for the van der Pol equation with $\varepsilon = 1000$, which is considered to be a “stiff” equation, is between 10^{4}–10^{5} [9]. Figure 1 compares the value of the temperature and stiffness index of the simulation. Figures 1a and 1c show the temperature and stiffness index respectively of a method of order 1 across the full 0.5 second range of the simulation. Figures 1b and 1d also show the temperature and stiffness index respectively of a method of order 1 but are stretched from 0.1 to 0.15 seconds of the simulation to show greater detail immediately before and after autoignition. Figure 1e demonstrates the reduction in stiffness due to an increase in order of the method to order 4.

The value of the stiffness index at the start of the evaluation was approximately 10^2. The stiffness index reached a minimum value of slightly less than 1 right as autoignition began. During the residence time, the index decreased by one order of magnitude. After spiking at its minimum value of less than 1, the stiffness index rapidly increased until reaching values between 10^7–10^{10}. The stiffness index values obtained have visually apparent noise after reaching the maximum values, even with a step size as small as 10^{-7}. Although further refinement of the step size and integration tolerance may reduce the magnitude of this noise to some extent, the values remained relatively consistent within the range of 10^7–10^{10}.

Furthermore, the stiffness index decreased substantially when the order of the method was increased to 4. While the stiffness index exhibited many of the same trends as before, the maximum stiffness index ranged between 10^3–10^4.

4. Conclusions

Our analysis demonstrated that the stiffness of the model prior to autoignition increases significantly during and after autoignition. This observation matches the expected behavior based on our intuition of the problem, but is based on a calculated metric. It also demonstrates the feasibility of using an explicit method prior to autoignition to save computational expense. Interestingly, we did not observe the maximum value of stiffness index during ignition; instead, this value occurred around 0.02 s after the reaching equilibrium (i.e., steady-state temperature and mass fractions). After reaching its maximum value, the stiffness remained relatively constant. The stiffness index evaluated provides a reasonable means of detecting the stiffness of combustion reactions. Incorporating calculation of this index in the code used to integrate the derivative vector will both accelerate calculation of the index and enable the index to be calculated on-the-fly.

5. Acknowledgements

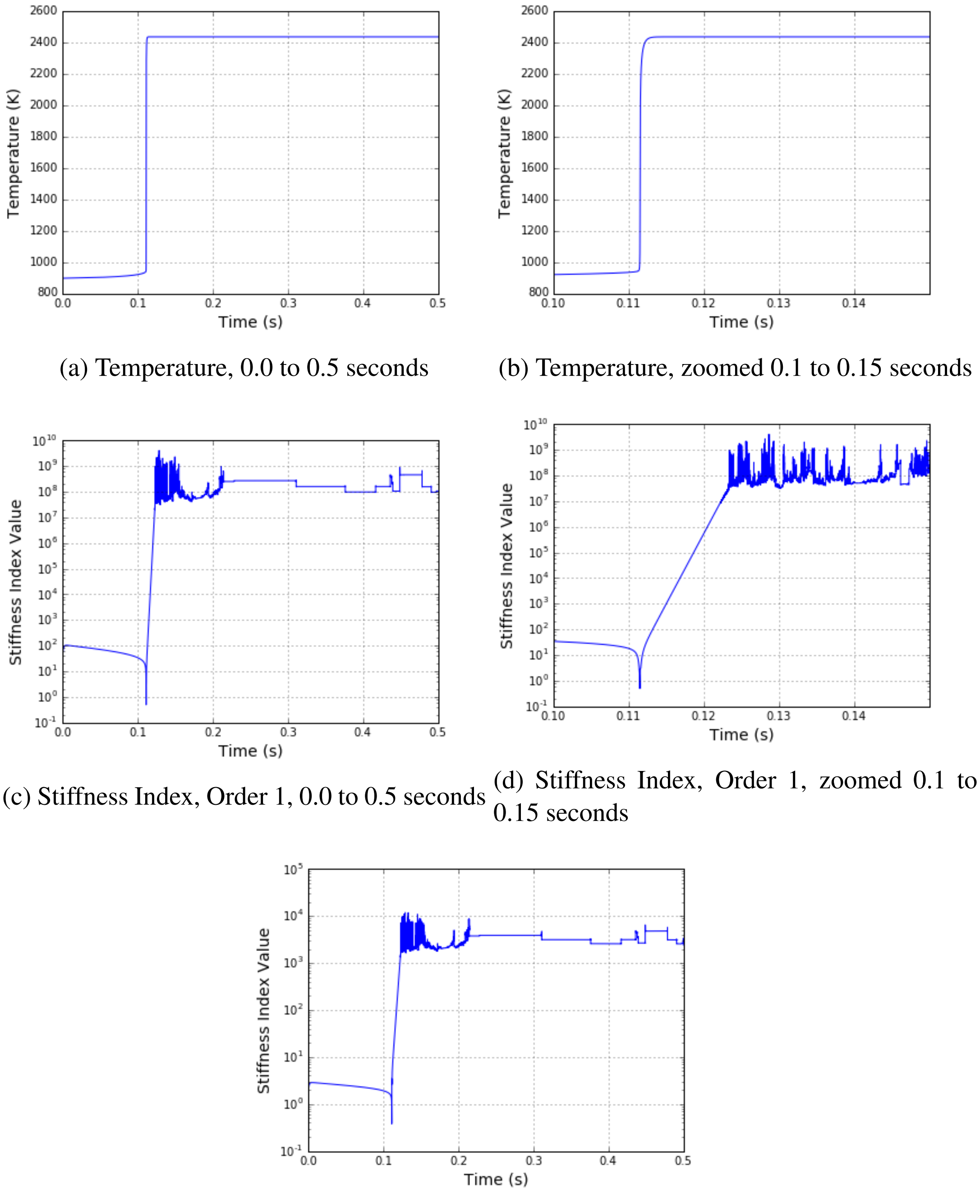
This material is based upon work supported by the National Science Foundation under grant ACI-1535065.

References

[1] A. H. Epstein, Aircraft engines' needs from combustion science and engineering, Combustion and Flame 159 (2012) 1791–1792. DOI: 10.1016/j.combustflame.2012.02.022.

[2] K. E. Niemeyer and O. State, Collaborative Research : SI2-SSE : An intelligent and adaptive parallel CPU / GPU co-processing software library for accelerating reactive-flow simulations, (2015).

[3] A. Trouvé, D. C. Haworth, J. H. Miller, L. K. Su, and A. Violi, *Cyber-Based Combustion Science: Report on the NSF Workshop on Cyber-Based Combustion Science*, National Science Foundation, Available at http://www-personal.umich.edu/~avioli/pdf/nsf-report.pdf, 2006.



(a) Temperature, 0.0 to 0.5 seconds

(b) Temperature, zoomed 0.1 to 0.15 seconds

(c) Stiffness Index, Order 1, 0.0 to 0.5 seconds

(d) Stiffness Index, Order 1, zoomed 0.1 to 0.15 seconds

(e) Stiffness Index, Order 4, 0.0 to 0.5 seconds

Figure 1: Temperature and stiffness index of the reaction calculated with a constant step size of 10^{-7} s. Note that the stiffness index reached its maximum values after 0.12 s when the temperature of the reaction had already stabilized.

[4] U.S. Department of Energy, *Basic Research Needs for Clean and Efficient Combustion of 21st Century Transportation Fuels*, Available at `http://science.energy.gov/~/media/bes/pdf/reports/files/ctf_rpt.pdf`, 2007.

[5] National Research Council, Transforming Combustion Research through Cyberinfrastructure, Available at `http://www.nap.edu/catalog.php?record_id=13049`, The National Academies Press, Washington, DC, 2011.

[6] National Research Council, Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science in 2017–2020, Available at `http://www.nap.edu/catalog.php?record_id=18972`, The National Academies Press, Washington, DC, 2014.

[7] T. Lu and C. K. Law, Toward accommodating realistic fuel chemistry in large-scale computations, Progress in Energy and Combustion Science 35 (2009) 192–215. DOI: `10.1016/j.pecs.2008.10.002`.

[8] G. Söderlind, L. Jay, and M. Calvo, Stiffness 1952-2012: Sixty years in search of a definition, BIT Numerical Mathematics 55 (2014) 531–558.

[9] L. F. Shampine, Measuring Stiffness, Applied Numerical Mathematics 1 (1985) 107–119. DOI: `10.1016/0168-9274(85)90020-0`.

[10] L. Brugnano, F. Mazzia, and D. Trigiante, Recent Advances in Computational and Applied Mathematics, Recent Advances in Computational and Applied Mathematics (2011) 1–21, arXiv: `0910.3780`.

[11] J. D. (D. Lambert, Computational methods in ordinary differential equations, Wiley, 1973, p. 278.

[12] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II, vol. 14, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[13] L. F. Shampine, Type-insensitive ODE codes based on implicit A-stable formulas, Mathematics of Computation 39 (1982) 109–123. DOI: `10.1090/S0025-5718-1982-0658216-2`.

[14] M. P. Burke, M. Chaos, Y. Ju, F. L. Dryer, and S. J. Klippenstein, Comprehensive H2/O2 kinetic model for high-pressure combustion, International Journal of Chemical Kinetics 44 (2012) 444–474. DOI: `10.1002/kin.20603`.

[15] K. E. Niemeyer, Reducing the Cost of Chemistry in Reactive-Flow Simulations: Novel Mechanism Reduction Strategies and Acceleration via Graphics Processing Units, PhD thesis, Cleveland, OH: Case Western Reserve University, 2013.

[16] K. E. Niemeyer, N. J. Curtis, and C.-J. Sung, `pyJac`: analytical Jacobian generator for chemical kinetics, Computer Physics Communications (2017), DOI: `10.1016/j.cpc.2017.02.004`.

[17] K. E. Niemeyer and N. J. Curtis, *pyJac v1.0.2*, 2017, DOI: `10.5281/zenodo.251144`.

[18] D. G. Goodwin, H. K. Moffat, and R. L. Speth, *Cantera: An Object-oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes*, `http://www.cantera.org`, Version 2.3.0, 2017, DOI: `10.5281/zenodo.170284`.

[19] E. Jones, T. Oliphant, P. Peterson, et al., *SciPy: Open source scientific tools for Python*, [Online; accessed 2017-02-28], 2001–, URL: `http://www.scipy.org/`.

[20] A. C. Hindmarsh, ODEPACK, A Systematized Collection of ODE Solvers." In Scientific Computing, IMACS Transactions on Scientific Computation 1 (1983) 55–64.

[21] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, The NumPy array: A structure for efficient numerical computation, Computing in Science and Engineering 13 (2011) 22–30. DOI: `10.1109/MCSE.2011.37`, arXiv: `1102.1523`.