Generalized preconditioning for accelerating simulations with large kinetic models

Anthony S. Walker[a], Raymond L. Speth[b], and Kyle E. Niemeyer[a]

[a] *School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, Oregon, United States*

[b] *Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, United States*

Abstract

Detailed modeling of the combustion of real transportation fuels and the atmospheric reactions involving their emissions is prohibitively expensive, due to the large size and stiffness of the kinetic models. Adaptive preconditioning is a method used to reduce the cost of integrating large kinetic models by forming a preconditioner based on an semi-analytical Jacobian matrix, paired with sparse linear algebra procedures. In this study, we extend this preconditioning method to a more-general mole-based state vector formulation applicable to generic reactor types and combinations. We tested the scheme using constant-pressure ideal-gas reactor simulations, showing speed-ups in performance from 0.46 to 1432 for chemical kinetic models with 10 to 7171 species. Overall, this method improves performance by more than three orders of magnitude for large kinetic models, and offers benefits for models with more than 100 species.

Keywords: Chemical kinetics; Implicit integrators; Sparse matrix; Preconditioner; Ordinary differential equations

1. Introduction

Combustion remains the driving force behind transportation, critical to the world economy and supply chain. While a necessity, transportation accounts for substantial amounts of greenhouse gas and pollutant emissions that adversely affect the environment and public health [1, 2]. Beyond carbon dioxide, the emissions produced consist of many additional chemical species, including pollutants such as carbon monoxide, volatile organic compounds, sulphur oxides, and nitrogen oxides [1]. Detailed modeling of the combustion of real transportation fuels and the secondary atmospheric reactions involving their emissions is prohibitively expensive, due to the large sizes of the kinetic models. For example, with large kinetic models of secondary organic aerosols containing in excess of 5000 species [3]. However, simulating and then mitigating the impact of these emissions requires the ability to model such complex phenomena on a practical scale.

Most approaches to including detailed chemistry in large-scale reacting-flow simulations currently use methods that either reduce the size of the model, via skeletal reduction or lumping, or eliminate short timescales to reduce stiffness [4, 5]. However, even when combined, these methods may not be sufficient to tackle the extremely large (and ever-growing) size of detailed kinetic models for real-fuel combustion and aerosol formation. Thus, accommodating large kinetic models in simulations also requires improvements in integration methods. The ordinary differential equations (ODEs) for chemical kinetics are numerically stiff and require implicit integration schemes, which can be particularly expensive for large models because they require the factorization of large matrices. Direct solutions of large linear systems are prohibitively expensive and limit large-scale simulations with operations that scale cubically with number of species in the kinetic model.

In this study, we extend a preconditioning method developed for constant-volume systems [6] to more-general problems and implement it in the open-source library `Cantera` [7]. We apply our extended method on ideal gas, constant-pressure reactor simulations, with the goal of obtaining substantial speed-up. This method, adaptive preconditioning, uses a semi-analytical Jacobian that makes several simplifications to accelerate its formulation. Notably, the species-rate partial derivatives neglect third-body efficiencies and pressure dependence, and the the temperature partial derivatives are approximated with finite differences. Furthermore, the "adaptive" feature of the method prunes non-diagonal values when below a specified threshold to further increase sparsity.

Leveraging sparsity has been a strategy for integrating systems of ODEs for quite some time [8, 9]. More recently, these strategies have been applied to combustion applications [10] and remain a popular method for improving performance. Others have used analytical Jacobian matrices in lieu of approximate finite-difference Jacobians, which can improve performance by only calculating non-zero matrix elements. Perini et al. [11] developed exact and approximate analytical Jacobian formulations for gas kinetics that leverage sparsity, implemented in a tool called `SpeedCHEM`. They applied these methods with various integration strategies, including preconditioning and incomplete lower-upper (LU) factorization [12]. Later, Perini et al. [13] used approximate exponential terms and logarithms to further speed up the integration of kinetic models. Dijkmans et al. [14] used an analytical Jacobian with in situ adaptive tabulation, coupled with efficient routines on graphics processing unit, to accelerate simulations with detailed chemistry. Similarly, Niemeyer et al. [15] developed `pyJac`, an analytical Jacobian generator for chemical kinetics that provides speed-up over finite-difference Jacobian calculations. Anzt et al. [16] leveraged sparsity by considering graphics processing unit-based Jacobi iteration and incomplete factorization preconditioning. Most recently, Lapointe et al. [17, 18] applied adaptive preconditioning to one-dimensional laminar flame simulations and flamelet calculations.

Here, we extend the adaptive preconditioning method to a generalized formulation using a mole-based state vector that is applicable to generic reactor types and combinations of reactors. We analyze how differences in the generalized formulation affect the structure of the preconditioner. To determine the impact of tunable parameters, we calculate ignition delays using both a standard direct solver and the preconditioned method with a range of parameter values and kinetic models, and evaluate the performance benefit of the preconditioned method. On the basis of these results, we develop recommendations or heuristics for solver and parameter selection and discuss directions for future work.

2. Methods

Nearly all numerical combustion codes integrate chemical kinetics using implicit methods to handle the stiffness. Schemes based on backwards differentiation formula (BDF) are commonly used; here, we focus on the `CVODES` solver from the `Sundials` package [19, 20]. The BDF method in `CVODES` relies on a nonlinear equation solver based on Newton iteration, which requires several linear iterations per nonlinear step. We use a left-preconditioned Generalized Minimum Residuals (`GMRES`) [21] method for the linear iterations, which solves a preconditioned system of linear equations. The solution of the preconditioned system is determined via an incomplete LU decomposition.

First, we set up the system by defining a state vector of length $\ell = \Psi + 1$, where Ψ is the number of chemical species. We represent this system with the state vector

$$\mathcal{S} = \{T, n_1, n_2, n_3, \cdots, n_\Psi\}, \quad \mathcal{S} \in \mathbb{R}^\ell, \quad (2.1)$$

where T is temperature and n_i is the moles of species i. A general system of governing ODEs is

$$\frac{d\mathcal{S}}{dt} = \mathcal{F}(t, \mathcal{S}) \ , \tag{2.2}$$

where $\mathcal{F}(t, \mathcal{S})$ can be a nonlinear function of t and $\mathcal{S}$.

Here, we extend the preconditioning strategy to an ideal-gas, constant-pressure reactor system. The mole-based species conservation equation for the i^{th} species is

$$\frac{dn_i}{dt} = V\dot{\omega}_i \ , \tag{2.3}$$

where $\dot{\omega}_i$ is the net rate of production. The mole-based conservation of energy equation is

$$\frac{dT}{dt} = \frac{-\sum_{k=1}^{\Psi} \bar{h}_k \dot{\omega}_k}{\sum_{k=1}^{\Psi} [\mathrm{n}_k] \hat{c}_{p,k}} = \frac{-\sum_{k=1}^{\Psi} \bar{h}_k \dot{n}_k}{\sum_{k=1}^{\Psi} n_k \hat{c}_{p,k}} \ , \tag{2.4}$$

where $\hat{c}_{p,k}$ and $\bar{h}_k$ are the molar constant pressure specific heat and enthalpy of species n_k, respectively.

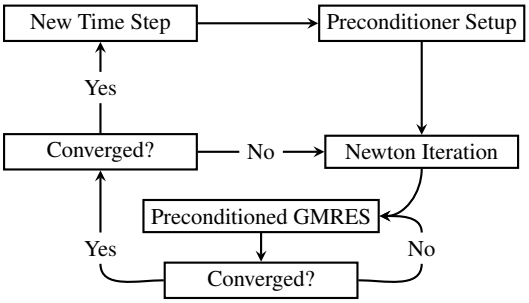

Fig. 1: General procedure of a time step using the preconditioned solver.

Figure 1 shows the process of integrating one time step, starting with setting up the preconditioner. The nonlinear solver then starts to make iterations which requires iterations of the linear solver. In some cases, the preconditioner is reformed during the nonlinear iterations as necessary.

2.1. Adaptive preconditioning

The adaptive preconditioning method originally applied to constant-volume systems by McNenly et al.[6] uses a mass-fraction-based approximate Jacobian matrix, which results in a fully dense system for constant-pressure applications. To generalize this method, we use a mole-based state vector and Jacobian matrix, $\mathcal{J}(\mathcal{S})$, to develop the preconditioner, $\mathcal{P}$. This approximate Jacobian is developed by eliminating elements below a certain threshold, computing finite difference temperature derivatives, and neglecting third-body efficiency and pressure dependence in the analytical derivatives $\frac{\partial \dot{n}_i}{\partial n_j}$. The approximate Jacobian matrix used in this work is

$$\mathcal{J}(\mathcal{S}) = \begin{bmatrix} \frac{\partial \dot{T}}{\partial T} & \frac{\partial \dot{T}}{\partial n_1} & \cdots & \frac{\partial \dot{T}}{\partial n_\Psi} \\ \frac{\partial \dot{n}_1}{\partial T} & \frac{\partial \dot{n}_1}{\partial n_1} & \cdots & \frac{\partial \dot{n}_1}{\partial n_\Psi} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \dot{n}_\Psi}{\partial T} & \frac{\partial \dot{n}_\Psi}{\partial n_1} & \cdots & \frac{\partial \dot{n}_\Psi}{\partial n_\Psi} \end{bmatrix} \ . \tag{2.5}$$

We first consider the derivatives with respect to temperature that are approximated with finite differences. We generalize them as $\frac{\partial \dot{\eta}}{\partial T}$, where η is any state variable and ϵ is a perturbation, such as machine precision (e.g., `DBL_EPSILON` in C++):

$$\frac{\partial \dot{\eta}}{\partial T} = \frac{\dot{\eta}\big|_{T+\sqrt{\epsilon}} - \dot{\eta}\big|_{T}}{\sqrt{\epsilon}} \ . \tag{2.6}$$

The derivatives of Eq. (2.3), $\frac{\partial \dot{n}_i}{\partial n_j}$, require looping over the number of species Ψ and number of reactions Φ. Since the preconditioner requires that we use moles, we must also convert any concentration-based terms to moles:

$$[\mathrm{n}_j] = \frac{n_j}{V} \tag{2.7}$$

$$\frac{\partial [\mathrm{n}_j]}{\partial n_j} = \frac{1}{V} \ , \tag{2.8}$$

where V is the volume of the system. The species-to-species partial derivatives used in the preconditioner are then

$$\frac{\partial \dot{n}_i}{\partial n_j} = \frac{\partial V}{\partial n_j}\dot{\omega}_i + V\frac{\partial \dot{\omega}}{\partial n_j} \ , \tag{2.9}$$

where $\dot{\omega}_i$ is the rate of production of species i. We break down Eq. 2.9 into components to determine the overall terms in a piece-wise manner. We first consider the derivative of the rate law with respect to moles:

$$\begin{aligned} V\frac{\partial \dot{\omega}_i}{\partial n_j} &= V\frac{\partial \dot{\omega}_i}{\partial [\mathrm{n}_j]}\frac{\partial [\mathrm{n}_j]}{\partial n_j} \\ &= \sum_{k=1}^{\Phi} \nu_{i,k} b_k \frac{\partial R_k}{\partial [\mathrm{n}_j]} + \nu_{i,k} R_k \frac{\partial b_k}{\partial [\mathrm{n}_j]} \ , \end{aligned} \tag{2.10}$$

where b_k is always one, ν represents the stoichiometric coefficients, and the volume terms cancel. Since b_k is always one, $\frac{\partial b_k}{\partial [\mathrm{n}_j]} = 0$ and Equation (2.10) then simplifies to

$$\frac{\partial \dot{\omega}_i}{\partial n_j} = \sum_{k=1}^{\Phi} \nu_{i,k} \frac{\partial R_k}{\partial [\mathrm{n}_j]} \ . \tag{2.11}$$

We continue to break down the rate of progress

derivatives into reactant and product contributions as

$$\frac{\partial R_k}{\partial [\mathrm{n}_j]} = k_{f,k} v'_{k,j} [\mathrm{n}_j]^{v'_{k,j}-1} \prod_{\substack{h=1 \\ h\neq j}}^{\alpha} [\mathrm{n}_h]^{v'_{k,h}} - k_{r,k} v''_{k,j} [\mathrm{n}_j]^{v''_{k,j}-1} \prod_{\substack{h=1 \\ h\neq j}}^{\beta} [\mathrm{n}_h]^{v''_{k,h}} , \quad (2.12)$$

where k is the reaction rate and α and β represent reactant and product species for the reaction. To determine $\frac{\partial V}{\partial n_j}\omega_i$, we start with the ideal gas law in the form

$$V = \frac{NRT}{P} . \quad (2.13)$$

We use this relation to obtain

$$\frac{\partial V}{\partial n_j} = \frac{\partial V}{\partial N}\frac{\partial N}{\partial n_j} = \frac{RT}{P} = \frac{V}{N} . \quad (2.14)$$

The final expression is then

$$\frac{\partial \dot{n}_i}{\partial n_j} = \frac{V}{N}\omega_i + \sum_{k=1}^{\Phi} \nu_{i,k} k_{f,k} v'_{k,j} [\mathrm{n}_j]^{v'_{k,j}-1} \prod_{\substack{h=1 \\ h\neq j}}^{\alpha} [\mathrm{n}_h]^{v'_{k,h}} + \sum_{k=1}^{\Phi} -\nu_{i,k} k_{r,k} v''_{k,j} [\mathrm{n}_j]^{v''_{k,j}-1} \prod_{\substack{h=1 \\ h\neq j}}^{\beta} [\mathrm{n}_h]^{v''_{k,h}} . \quad (2.15)$$

The derivatives of temperature with respect to different species, $\frac{\partial \dot{T}}{\partial n_j}$, are from Eq. (2.4), where we represent the numerator as a and the denominator as b. We find the derivatives by first applying the quotient rule:

$$\frac{\partial \dot{T}}{\partial n_j} = \frac{a'b - ab'}{b^2} . \quad (2.16)$$

The terms in the former equation are then found as

$$a'b = -N\hat{c}_p \sum_{k=1}^{\Psi} \bar{h}_k \frac{\partial \dot{n}_k}{\partial n_j} \quad (2.17)$$

and

$$ab' = -c_{p,j} \sum_{k=1}^{\Psi} \bar{h}_k \dot{n}_k . \quad (2.18)$$

The $a'b$ term actually contains existing Jacobian elements, so in practice we pre-calculate these elements and reuse them here. Finally, we combine the two parts into the generalized temperature derivative with respect to species j:

$$\frac{\partial \dot{T}}{\partial n_j} = \frac{-N\hat{c}_p \sum_{k=1}^{\Psi} \bar{h}_k \frac{\partial \dot{n}_k}{\partial n_j} + c_{p,j} \sum_{k=1}^{\Psi} \bar{h}_k \dot{n}_k}{(N\hat{c}_p)^2} . \quad (2.19)$$

The preconditioner is then formed as

$$\mathcal{P} = I - \gamma \mathcal{J}(\mathcal{S}) , \quad (2.20)$$

where

$$\gamma = \Delta t \beta \quad (2.21)$$

and β is a coefficient in the BDF formulation used to obtain a particular order. The final step in $\mathcal{P}$ formation is removing non-diagonal elements below a threshold value, ξ:

$$\mathcal{P}[i,j] = \begin{cases} 0 & \mathcal{P}[i,j] < \xi \text{ and } i \neq j, \\ \mathcal{P}[i,j] & i = j \text{ or } \mathcal{P}[i,j] \geq \xi . \end{cases} \quad (2.22)$$

2.2. Newton iteration

The nonlinear portion of the integration was performed with Newton's method, which linearly approximates the nonlinear system using Taylor series expansion. `Sundials` does this internally, but discussion is useful to show how the preconditioner is used during integration [20]. We started this process by discretizing the time integration of Eq. (2.2) using a general, variable-order, BDF:

$$\sum_{i=0}^{\Omega} \alpha^{n-i} \mathcal{S}^{n-i} = \beta \Delta t \mathcal{F}(t, \mathcal{S}_n) , \quad (2.23)$$

where Ω is the order, n is the current time step, and α^k and β are coefficients used to achieve the order. For example, in the case of backward Euler (i.e., first-order implicit scheme), $\Omega = 1$, $\alpha_0 = 1$, $\alpha_1 = -1$, and $\beta = 1$. We put the formula in the form

$$\sum_{i=0}^{\Omega} \alpha^{n-i} \mathcal{S}^{n-i} - \beta \Delta t \mathcal{F}(t, \mathcal{S}^n) = 0 , \quad (2.24)$$

which is a system of nonlinear algebraic equations. We use the system

$$\mathcal{N}(\mathcal{S}_n) \equiv \sum_{i=0}^{\Omega} \alpha^{n-i} \mathcal{S}^{n-i} - \beta \Delta t \mathcal{F}(t, \mathcal{S}^n) = 0, \quad (2.25)$$

and apply a Taylor series approximation to obtain

$$\mathcal{N}(\mathcal{S}) = \mathcal{N}(\mathcal{S}^{n-1}) + \mathcal{J}(\mathcal{N}(\mathcal{S}^{n-1}))(\mathcal{S}^n - \mathcal{S}^{n-1}) + \text{higher order terms} = 0 . \quad (2.26)$$

We truncate the series (2.26) after the linear term to provide a linear system of equations with iteration index k as

$$\mathcal{J}(\mathcal{N}(\mathcal{S}^k))(\mathcal{S}^{k+1} - \mathcal{S}^k) = -\mathcal{N}(\mathcal{S}^k), \quad k \in \mathbb{R}_0^+ , \quad (2.27)$$

Model	Formula	Species	Reactions	Optimal Threshold (ξ)
Hydrogen [22]	H_2	10	29	0
GRI-Mech 3.0 [22]	CH_4	55	325	10^{-1}
DME-Propane [23]	CH_3OCH_3 & C_3H_8	122	711	10^{-8}
HyChem Jet-A [24, 25]	POSF 10325($C_{11}H_{22}$)	203	1589	10^{-10}
Butane [26]	C_4H_{10}	230	2461	0
n-Heptane [27]	*n*-C_7H_{16}	654	4846	10^{-4}
Isooctane [28]	*i*-C_8H_{18}	874	6864	0
3-Methylheptane [28]	$C_8H_{18}-3$	1378	8143	10^{-1}
n-Hexadecane [29]	*n*-$c_{16}h_{34}$	2115	13341	0
Methyl-5-decenoate [30]	MD_5D	2649	10487	10^{-4}
Methyl-decanoate & *n*-heptane [30]	MD & *n*-C_7H_{16}	3787	10264	10^{-11}
2-Methyl-nonadecane [31]	$C_{20}H_{42}-2$	7171	38324	10^{-3}

Table 1: Details on the chemical kinetic models used for testing.

which is iterated to convergence of a solution to the nonlinear system of equations for time step n. However, iteration of the linear system requires that we develop an iteration formula. The Jacobian of $\mathcal{N}(\mathcal{S}^k)$ is simplified to

$$\mathcal{K}(\mathcal{S}^k) \equiv \mathcal{J}(\mathcal{N}(\mathcal{S}^k)) = I - \Delta t \beta \mathcal{J}(\mathcal{F}(t, \mathcal{S}^k)) \ , \tag{2.28}$$

and substituted into the linear system to yield

$$\mathcal{K}(\mathcal{S}^k)(\mathcal{S}^{k+1} - \mathcal{S}^k) = -\mathcal{N}(\mathcal{S}^k) \ . \tag{2.29}$$

Finally, we form our iteration formula:

$$\mathcal{S}^{k+1} = \mathcal{S}^k - \left[\mathcal{K}(\mathcal{S}^k)\right]^{-1} \mathcal{N}(\mathcal{S}^k) \ . \tag{2.30}$$

2.3. Preconditioned GMRES

The nonlinear solution requires several linear iterations to complete a nonlinear iteration. In this work, we use a left-preconditioned `GMRES` method [21] to leverage the sparsity of large kinetic models. We consider the system

$$Ax = b, \quad A \in \mathbb{C}^{m \times n},\ m \geq n,\ b \in \mathbb{C}^m \tag{2.31}$$

to describe the method. A direct solution of (2.31) is

$$x = A^{-1} b \ . \tag{2.32}$$

However, when the problem is ill-conditioned it helps to precondition the system as

$$\mathcal{P}^{-1} A x = \mathcal{P}^{-1} b \ . \tag{2.33}$$

Ideally, the preconditioner, $\mathcal{P}$, would be close to A and well-posed, to accelerate convergence to the solution. If $\mathcal{P} = A$, then the solution would be obtained exactly from preconditioning. Applying preconditioning to the `GMRES` method, we define the residual as

$$\delta = ||\mathcal{P}^{-1} b - \mathcal{P}^{-1} A x||_2 \tag{2.34}$$

and attempt to find x such that δ is minimized. Translating this idea, we put our system in the form

$$\mathcal{K}(\mathcal{S}^k)\mathcal{S}^{k+1} = \mathcal{K}(\mathcal{S}^k)\mathcal{S}^k - \mathcal{N}(\mathcal{S}^k) \tag{2.35}$$

and define the right-hand side as $\mathcal{R}(\mathcal{S}^k)$ to get

$$\mathcal{K}(\mathcal{S}^k)\mathcal{S}^{k+1} = \mathcal{R}(\mathcal{S}^k) \ . \tag{2.36}$$

Applying the preconditioner to the system, we obtain

$$\mathcal{P}(\mathcal{S}^k)^{-1}\mathcal{K}(\mathcal{S}^k)\mathcal{S}^{k+1} = \mathcal{P}(\mathcal{S}^k)^{-1}\mathcal{R}(\mathcal{S}^k) \ , \tag{2.37}$$

which can be substituted into Eq. (2.34) to form a final expression for the residual.

We applied the preconditioned `GMRES` method in `CVODES` by a user-defined function, `PSolve`, which requires that the user solves the system

$$\mathcal{P} z = r \ , \tag{2.38}$$

where r in this case is equivalent to $b - Ax$. The result is the preconditioned residual in the output vector z:

$$z = \mathcal{P}^{-1}(b - Ax) \ . \tag{2.39}$$

Here, we applied the incomplete LU decomposition from the `Eigen` C++ package [32] to determine the solution to the former system. The two-norm of z is then the residual δ. Applying the `CVODES` method to our system, we obtained

$$\begin{aligned} z &= \mathcal{P}(\mathcal{S}^k)^{-1}\left[\mathcal{R}(\mathcal{S}^k) - \mathcal{K}(\mathcal{S}^k)\mathcal{S}^{k+1}\right] \\ &= \mathcal{P}(\mathcal{S}^k)^{-1}\mathcal{B}(\mathcal{S}^k) \ , \end{aligned} \tag{2.40}$$

where the preconditioner is formed with adaptive preconditioning described in Section 3.1.

2.4. Summary of method

To summarize the integration procedure, we applied adaptive preconditioning to a constant-pressure ideal gas reactor. The integration method is an implicit BDF formulation that uses Newton iterations

for the nonlinear solution and preconditioned GMRES for the linear solution. The preconditioned solution used by GMRES is formed with an incomplete LU factorization. We implemented these methods within Cantera and take advantage of Sundials [20] and Eigen [32].

3. Results & discussion

We implemented adaptive preconditioning in Cantera [7] to integrate a constant-pressure ideal-gas reactor system. We selected a set of 12 chemical kinetic models, spanning a range of species numbers from 10 to 7171, listed in Table 1. In this problem, we provided a stoichiometric fuel/air mixture at atmospheric pressure and an initial temperature of 1500 K. We selected these initial conditions to produce an ignition event and integrated to a time of 1 s. We integrated this problem with a range of preconditioner thresholds from 10^{-18} to 10^{-1}, as well as 0 (i.e., no threshold). We ran each test case 100 times and used average values to help account for any unknown variability in timing the runs. These test cases were executed using the Oregon State University College of Engineering High-Performance Computing Cluster; all code necessary to reproduce these results is available openly [33].

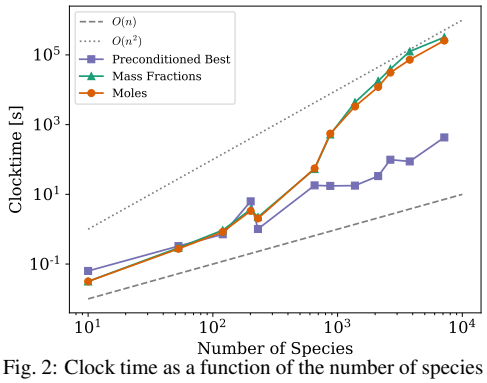

Fig. 2: Clock time as a function of the number of species for preconditioned and non-preconditioned systems.

Figure 2 shows the wall-clock time of ignition simulations as a function of the number of species in the kinetic model. The preconditioned solver begins showing improvement in clock time around one hundred species, while the baseline solver performs better for models with fewer than one hundred species. For the largest models, we obtained speed-up of several orders of magnitude, and see the most benefit as the number of species increases. Specifically, the maximum speed-up is 1432 for the methyl decanoate/*n*-heptane [30] model with 3787 species, and the minimum speed-up is 0.46 with the hydrogen [22] model at 10 species. Overall, though, the preconditioned solver offers significant improvement in performance for larger models, with a slight reduction in performance (at most just about half) for the smallest models.

The driving factor behind speed-up seems to be the number of species. However, the performance trend fluctuates with the DME-propane model at 122 species, where the average speed-up is 1.44, and the Jet A model at 201 species, where the average speed-up is 0.58. The butane model has 230 species, similar to the number of species in the Jet A model, but has an average speed-up of 2.2. This deviation in performance with the Jet A model suggests that there is more to performance gains than strictly the number of species.

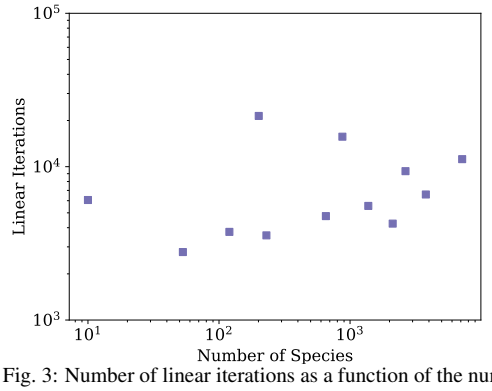

Fig. 3: Number of linear iterations as a function of the number of species.

Performance increases are consistently accompanied by a reduction in the number of linear iterations taken by the preconditioned solvers. Figure 3 shows that the number of linear iterations remains roughly constant for the preconditioned method. We see several spikes in the numbers of linear iterations, which are consistent with spikes in the number of falloff and/or third-body reactions in a given model. In particular, the Jet A model has more of these reactions types than any other tested model, which is accompanied by it also requiring the most linear iterations.

In contrast, the *n*-heptane model has a large number third-body and falloff reactions, but it also has more total species, so the solution is less impacted by these reactions being present. This trend is likewise carried through the rest of the results but more subtly because the impact is diluted by the number of species present. For example, we see a valley in the clock time of the larger models (i.e., MD_5D, MD & *n*-C_7H_{16}, and $C_{20}H_{42-2}$), which tracks well with the number of falloff/third-body and linear iterations in Fig. 3. The performance likely depends on the number of these reaction types since these features are neglected in the formation of the preconditioner, suggesting that further improvements could be achieved by better-handling these reactions.

Figure 4 shows the range of speed-up as a function of the threshold value for all models. The threshold

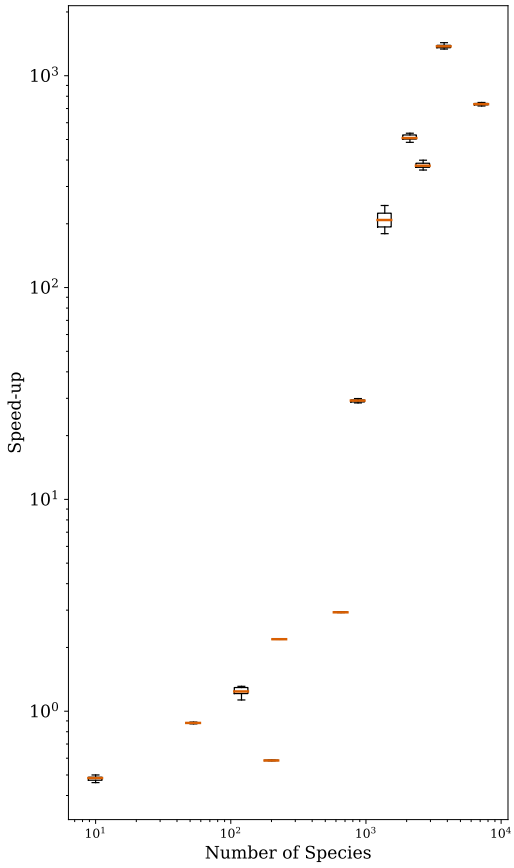

Fig. 4: Variation and median (orange line) speed-up of the preconditioned solver with changes in threshold (over 1×10^{-18} to 1×10^{-1} and 0).

minimally impacts the performance of the preconditioned solver, with variation in speed-up generally being an order of magnitude less than the speed-up itself. Likewise, the mode of the optimal thresholds listed in Table 1 is 0, further supporting the lack of impact from threshold variation.

4. Conclusions

In this work, we extended an adaptive preconditioning method for implicit ODE integrators. Adaptive preconditioning neglects pressure dependence and third-body efficiencies when forming the Jacobian matrix, uses finite differences for temperature derivatives, and prunes small preconditioner matrix elements based on a threshold value. This also enables the use of sparse linear algebra operations. We extended on the previously demonstrated adaptive preconditioning method to consider a state vector based on moles, allowing its use in more-general application such as constant-pressure reactors. We tested this preconditioner on constant-pressure ideal-gas reactor simulations for kinetic models with a large range of species counts and over a range of threshold values.

In this application, the extended preconditioning method speeds-up performance by a factor of 0.46 to 1432, for models with numbers of species from 10 to 7171. The potentially substantial performance gains of the solver strongly depend on the total number of species and the number of falloff/third-body reactions in the kinetic model. These interactions directly influence the number of linear iterations required by the solver, which drives performance benefits. We also found that the preconditioner's threshold has a relatively low impact on performance, at generally an order of magnitude lower than the performance increase itself. The low impact of the threshold is likely due to the use of incomplete LU factorization, which can produce LU factors that are much more sparse and dominate performance benefits. Consequently, we can likely omit the threshold used to prune the preconditioner without negatively impacting performance.

Future work will explore how to balance sparsity with better-capturing third-body and pressure-dependent reactions in the semi-analytical Jacobian formulation. In addition, we will extend this method to support reactors with surface reactions and networks of reactors.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. 1931592 (OSU) and 1931391 (MIT).

Supplementary material

The results and figures in this article can be reproduced using Cantera [7] and the testing package shared openly [33].

References

[1] Y. Van Fan, S. Perry, J. J. Klemeš, C. T. Lee, A review on air emissions assessment: Transportation, Journal of Cleaner Production 194 (2018) 673–684.

[2] I. Manisalidis, E. Stavropoulou, A. Stavropoulos, E. Bezirtzoglou, Environmental and health impacts of air pollution: a review, Frontiers in Public Health 8 (2020) 14.

[3] J. Li, M. Cleveland, L. D. Ziemba, R. J. Griffin, K. C. Barsanti, J. F. Pankow, Q. Ying, Modeling regional secondary organic aerosol using the Master Chemical Mechanism, Atmospheric Environment 102 (2015) 52–61.

[4] T. Lu, C. K. Law, Toward accommodating realistic fuel chemistry in large-scale computations, Progress in Energy and Combustion Science 35 (2) (2009) 192–215.

[5] P. Pepiot, L. Cai, H. Pitsch, Model reduction and lumping procedures, in: Computer Aided Chemical Engineering, Elsevier, 2019, pp. 799–827.

[6] M. J. McNenly, R. A. Whitesides, D. L. Flowers, Faster solvers for large kinetic mechanisms using adaptive preconditioners, Proceedings of the Combustion Institute 35 (1) (2015) 581–587.

[7] D. G. Goodwin, R. L. Speth, H. K. Moffat, B. W. Weber, Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes, https://www.cantera.org, version 2.5.1 (2021). doi:10.5281/zenodo.4527812.

[8] P. N. Brown, A. C. Hindmarsh, Reduced storage matrix methods in stiff ODE systems, Applied Mathematics and Computation 31 (1989) 40–91.

[9] Y. Saad, ILUT: A dual threshold incomplete lu factorization, Numerical Linear Algebra with Applications 1 (4) (1994) 387–402.

[10] Y. M. Marzouk, A. F. Ghoniem, H. N. Najm, Toward a flame embedding model for turbulent combustion simulation, AIAA Journal 41 (4) (2003) 641–652.

[11] F. Perini, E. Galligani, R. D. Reitz, An analytical Jacobian approach to sparse reaction kinetics for computationally efficient combustion modeling with large reaction mechanisms, Energy & Fuels 26 (8) (2012) 4804–4822.

[12] F. Perini, E. Galligani, R. D. Reitz, A study of direct and Krylov iterative sparse solver techniques to approach linear scaling of the integration of chemical kinetics with detailed combustion mechanisms, Combustion and Flame 161 (5) (2014) 1180–1195.

[13] F. Perini, R. D. Reitz, Fast approximations of exponential and logarithm functions combined with efficient storage/retrieval for combustion kinetics calculations, Combustion and Flame 194 (2018) 37–51.

[14] T. Dijkmans, C. M. Schietekat, K. M. Van Geem, G. B. Marin, GPU based simulation of reactive mixtures with detailed chemistry in combination with tabulation and an analytical Jacobian, Computers & Chemical Engineering 71 (2014) 521–531.

[15] K. E. Niemeyer, N. J. Curtis, C.-J. Sung, pyJac: Analytical Jacobian generator for chemical kinetics, Computer Physics Communications 215 (2017) 188–203.

[16] H. Anzt, M. Gates, J. Dongarra, M. Kreutzer, G. Wellein, M. Köhler, Preconditioned Krylov solvers on GPUs, Parallel Computing 68 (2017) 32–44.

[17] S. Lapointe, R. A. Whitesides, M. J. McNenly, Sparse, iterative simulation methods for one-dimensional laminar flames, Combustion and Flame 204 (2019) 23–32.

[18] S. Lapointe, Y. Xuan, H. Kwon, R. A. Whitesides, M. J. McNenly, A computationally-efficient method for flamelet calculations, Combustion and Flame 221 (2020) 94–102.

[19] S. D. Cohen, A. C. Hindmarsh, P. F. Dubois, CVODE, A Stiff/Nonstiff ODE Solver in C, Computers in Physics 10 (2) (1996) 138.

[20] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, C. S. Woodward, SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers, ACM Transactions on Mathematical Software (TOMS) 31 (3) (2005) 363–396.

[21] L. N. Trefethen, D. Bau III, Numerical linear algebra, Vol. 50, SIAM, 1997.

[22] G. P. Smith, D. M. Golden, M. Frenklach, N. W. Moriarty, B. Eiteneer, M. Goldenberg, C. T. Bowman, R. K. Hanson, S. Song, W. C. Gardiner Jr., V. V. Lissianski, Z. Qin, GRI-Mech 3.0, http://combustion.berkeley.edu/gri-mech (1999).

[23] E. E. Dames, A. S. Rosen, B. W. Weber, C. W. Gao, C.-J. Sung, W. H. Green, A detailed combined experimental and theoretical study on dimethyl ether/propane blended oxidation, Combustion and Flame 168 (2016) 310–330.

[24] H. Wang, R. Xu, K. Wang, C. T. Bowman, R. K. Hanson, D. F. Davidson, K. Brezinsky, F. N. Egolfopoulos, A physics-based approach to modeling real-fuel combustion chemistry-I. Evidence from experiments, and thermodynamic, chemical kinetic and statistical considerations, Combustion and Flame 193 (2018) 502–519.

[25] R. Xu, K. Wang, S. Banerjee, J. Shao, T. Parise, Y. Zhu, S. Wang, A. Movaghar, D. J. Lee, R. Zhao, A physics-based approach to modeling real-fuel combustion chemistry–II. Reaction kinetic models of jet and rocket fuels, Combustion and Flame 193 (2018) 520–537, publisher: Elsevier.

[26] J. Zhang, L. Pan, J. Mo, J. Gong, Z. Huang, C. K. Law, A shock tube and kinetic modeling study of n-butanal oxidation, Combustion and Flame 160 (9) (2013) 1541–1549.

[27] 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.

[28] M. Mehl, H. J. Curran, W. J. Pitz, C. K. Westbrook, Chemical kinetic modeling of component mixtures relevant to gasoline, Tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States) (2009).

[29] C. K. Westbrook, W. J. Pitz, O. Herbinet, E. J. Silke, H. J. Curran, A detailed chemical kinetic reaction mechanism for n-alkane hydrocarbons from n-octane to n-hexadecane, Tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States) (2007).

[30] O. Herbinet, W. J. Pitz, C. K. Westbrook, Detailed chemical kinetic mechanism for the oxidation of biodiesel fuels blend surrogate, Combustion and Flame 157 (5) (2010) 893–908.

[31] S. M. Sarathy, C. K. Westbrook, M. Mehl, W. J. Pitz, C. Togbe, P. Dagaut, H. Wang, M. A. Oehlschlaeger, U. Niemann, K. Seshadri, Comprehensive chemical kinetic modeling of the oxidation of 2-methylalkanes from C_7 to C_{20}, Combustion and flame 158 (12) (2011) 2338–2357.

[32] G. Guennebaud, B. Jacob, others, Eigen v3 (2010). URL http://eigen.tuxfamily.org

[33] A. S. Walker, cantera_adaptive_testing, https://github.com/anthony-walker/cantera-adaptive-testing (2022).