



## An improved stiff-ODE solving framework for reacting flow simulations with detailed chemistry in OpenFOAM

Kun Wu, Yuting Jiang, Zhijie Huo, Di Cheng & Xuejun Fan


To cite this article: Kun Wu, Yuting Jiang, Zhijie Huo, Di Cheng & Xuejun Fan (2023) An improved stiff-ODE solving framework for reacting flow simulations with detailed chemistry in OpenFOAM, Combustion Theory and Modelling, 27:1, 57-82, DOI: [10.1080/13647830.2022.2153739](https://doi.org/10.1080/13647830.2022.2153739)


To link to this article: <https://doi.org/10.1080/13647830.2022.2153739>

 View supplementary material 

 Published online: 12 Dec 2022.

 Submit your article to this journal 

 Article views: 179

 View related articles 

 View Crossmark data 



## An improved stiff-ODE solving framework for reacting flow simulations with detailed chemistry in OpenFOAM

Kun Wu<sup>a†\*</sup>, Yuting Jiang<sup>b†</sup>, Zhijie Huo<sup>a</sup>, Di Cheng<sup>a</sup> and Xuejun Fan<sup>a,c</sup>

<sup>a</sup>State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics, Chinese Academy of Sciences, Beijing, People's Republic of China; <sup>b</sup>Hefei Zhongke Chongming Technology Company, Hefei, People's Republic of China; <sup>c</sup>School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing, People's Republic of China

(Received 25 March 2022; accepted 11 November 2022)

The integration of stiff ordinary differential equation (ODE) systems associated with detailed chemical kinetics is computationally demanding in practical combustion simulations. Despite the various approaches in expediting the computational efficiency, it is still necessary to optimise the cell-wise calculation in operator-splitting type simulations of reactive flow. In this work, we proposed an improved stiff-ODE solver framework targeting to speed up the simulation of reactive flow in OpenFOAM. This framework combines the Radau-IIA and backward differentiation formula (BDF) ODE-integration algorithms, the pyJac-based fully analytical Jacobian formulation, and dense-based LAPACK and sparse-based KLU sophisticated linear system solvers. We evaluate the performance of the efficient solver framework on various benchmark combustion problems across a wide range of chemical kinetic complexities. A comprehensive investigation of the key elements of stiff ODE solvers is conducted in the homogeneous reactor, focusing respectively on the influences of error tolerance, integration time interval, Jacobian evaluation methodology, and linear system solver on the accuracy and efficiency trade-off. More realistic simulation results are presented regarding the one-dimensional laminar flame and three-dimensional turbulent flame. The results indicate that the Radau-IIA is more preferable in both efficiency and accuracy compared with the widely used BDF and Seulex methods for large integration interval, whereas the differences between three methods diminish as the integration time interval decreases. In all cases, it is found that the full analytical Jacobian is more advantageous for small mechanisms of species number around 50–100 while the approximated formulation of Jacobian is recommended for larger ones. Furthermore, the more robust linear system solvers provide significant improvement on computational efficiency with the dense-based LAPACK solver being more suitable for small to moderate-scale mechanisms while sparse-based KLU being superior for large-scale mechanisms. The proposed efficient solver framework in its optimal configuration obtains more than 2.6 times speedup in realistic high-fidelity flame simulation with a 57 species combustion mechanism.

**Keywords:** reacting flow simulation; detailed chemistry; stiff ODE; Jacobian; linear system solver

---

\*Corresponding author. Email: [wukun@imech.ac.cn](mailto:wukun@imech.ac.cn)

†These authors contributed equally to this work.

## 1. Introduction

Recently, accurate and affordable numerically predictive tools for chemical reactive flows become indispensable in the design and optimisation of combustion devices [1]. Commonly, numerical simulations of reactive flows employ an operator-splitting approach to solve chemistry in a fractional step within which the temporal evolution of composition is only dependent on the local thermochemical state. Those chemistries can be described by stiff and nonlinear ordinary differential equations (ODEs), where the stiffness stems from the huge disparities between the characteristic chemistry timescales of different reactive species [2]. Correspondingly, in high-fidelity simulations involving finite rate chemistry, chemistry evaluation comprises the most computationally demanding part of the simulations [3]. Despite the development of techniques to accelerate the computation such as the chemistry mechanism reduction or dynamic adaptive chemistry [4,5], storage/retrieval tabulation [6,7], cell-clustering method [8,9], hardware-based acceleration [10], and combinations of these approaches [11], all these acceleration strategies still require accurate yet efficient stiff ODE solvers tailored for cell-wise calculations.

To be specific, in finite-rate chemistry approach, the computational cost associated with the chemistry solution originated from three major aspects, including the ODE-integration, Jacobian evaluation, and linear system solution. The numerical algorithm of an ODE integrator decisively determines the order of accuracy and computational efficiency. Owing to the better efficiency and stability, chemistry systems are commonly solved using implicit integration methods, among which the multi-step, backward differentiation formulation (BDF) method [12,13] is the most widely used one for combustion simulations [14]. Despite its high-order of accuracy, BDF method suffers from the re-initialisation problem and is prone to accumulation error, since it always starts with low-order approximation and builds up higher order successively [15,16]. To alleviate the concentration of computational cost at the beginning of every iteration, Imren and Haworth [15] resorted to an extrapolation-based method namely Seulex. The Seulex algorithm applies semi-implicit Euler method on the recursively partitioned sub-intervals and successively improves the solution accuracy through high-order projection [17]. As a one-step method, Seulex requires less information to reinitialise integration and achieves better performance than BDF method [15].

As it stands, the Seulex method is also a variant-order method like BDF, while the cost of reaching high order of accuracy through the increasing number of sub-intervals would somehow become a bottleneck for highly stiffed chemistry ODEs. These deficiencies inspire the exploitation of the implicit Runge–Kutta method (IRK) with fixed-order of accuracy. With this type of algorithm, the startup overhead is lower and the maximum order of accuracy is available immediately on the first iteration with a relatively larger internal time step. However, to the authors' knowledge, only a few showcases have been made with relatively simple configurations [16,18], and applications of the IRK method in reactive flow simulations associated with detailed chemistry still remain insufficient.

Generally, the implicit integration algorithms form non-linear systems whose roots are found iteratively by variant Newton methods hence systems of linear algebraic equations have to be solved. Nevertheless, OpenFOAM's native stiff ODE solvers employ basic method of LU decomposition, which is generally expensive because the computational cost scales cubically with respect to the size of the system [19]. To expedite the matrix operation, Morev et al. [20] made use of the standard linear algebra library (LAPACK) and gained an order of magnitude calculation speedup for small-scale chemistries. However,

with dense matrix representation, their implementation is more suitable for solving small matrix ( $< 500 \times 500$ ) and its performance may deteriorate for large and sparse matrix associated with detailed chemistries. On the contrary, for large-scale chemistry systems, sparse representation was found to be more efficient than the dense one, whose computational cost could achieve linear dependence on the size of the system [21]. Unfortunately, Morev et al.'s observations were made based on two relatively small-scale skeletal mechanisms comprising 21 and 54 species respectively, whereby the high-performance linear system solvers leveraging sparse matrix algebra, are still lacking.

As for the evaluation of Jacobian matrix  $\mathcal{J}$  involved in the Newton iteration, it contributes to another important part of the computational budget. Conventionally,  $\mathcal{J}$  is obtained by the finite differencing method, which may be an expensive operation depending on the scale of the chemical mechanisms. Lu et al. suggested that the computational cost of evaluating  $\mathcal{J}$  scales quadratically with respect to the number of reactions [2]. In order to cope with the increasing computational demand of these operation with the increase in mechanism size, a variant of chemical Jacobian approaches have been developed aiming at either preconditioning or approximating the chemical Jacobian matrix [14,21–23]. For steady-state reacting flows simulations, the diagonal approximation has been frequently invoked [24–26], which eliminates the expense of inverting the large block matrices that arise when species conservation equations are involved. This is accomplished by replacing the chemical Jacobian matrix by a diagonal matrix that is tailored to account for the fastest reactions in the chemical system. Nevertheless, they were argued to be inappropriate for time-accurate simulation of unsteady flows [25]. Many previous efforts have shown that sparse analytical Jacobian is more preferred when solving the ODE system with tight error tolerances [27–29].

The analytical Jacobian formulation exploits the functional forms of chemical reactions and thermodynamic properties to perform analytical differentiating. The constructing process is thus dependent on the treatment of different reaction types. Apparently, Jacobian formulation regarding elemental reaction dictated by Arrhenius kinetic law is relatively straightforward, whereas for more complex reaction behaviours such as third-body and pressure-dependent reactions, the evaluation of analytical Jacobian becomes challenging. In order to cope with the issue which leads to dense lines in the Jacobian matrix for each species participating in third-body reactions, several approximation approaches [14,21] were devised to ensure better computational efficiency and matrix sparsity. Specifically, in OpenFOAM version 7 [30], an approximation approach proposed by Schwer et al. [31] was implemented to maximise the computational efficiency of analytical Jacobian, by creating high sparsity at the expense of embedding numerical inaccuracies in the Jacobian. However, we have found that this approximation is more suitable for large-scale mechanisms with sparse Jacobians but fails to gain a fast solution for small- to medium-scale mechanisms featuring dense Jacobians. Consequently, the extent to which Jacobian evaluation formulation would influence the overall computational accuracy and efficiency is still elusive.

In the present work, we devise a new stiff chemistry solving methodology tailored for accurate yet efficient reactive flow simulations in OpenFOAM by targeting the aforementioned three major issues. First, an IRK integration algorithm, namely Radau-IIA, is implemented to improve the computational performance of the stiff ODE-integration algorithm in OpenFOAM. Second, we exploit standard linear algebra libraries, including both dense-based and sparse-based methods to further expedite the ODE calculation procedure by replacing the native LU decomposition and back substitution operating algorithm

in OpenFOAM standard library. Third, we introduce a fully analytical Jacobian (FAJ) formulation using the pyJac library, and demonstrate its efficacy in the fast calculation of chemistry problems and the trade-off between efficiency gain and numerical accuracy in various combustion layouts.

The remainder of this paper is arranged as follows. In Section 2, the numerical methodologies regarding the ODE-integration algorithms, the Jacobian evaluation formulations, and the linear system solvers as well as their numerical implementation are briefly presented. Then, we demonstrate the correctness and computational efficiency of the new integral chemistry solving framework against benchmark combustion problems and a well-established experimental configuration (Sandia Flame D) and discuss the implications of the results in Section 3. Finally, the conclusions are given in Section 4.

## 2. Numerical methodologies and their implementation

### 2.1. Principles of solving stiff ODEs

For combustion flow simulations using an operator-splitting method, the evolution of chemistry in the reacting fractional step is governed by the ODEs:

$$\frac{d\phi}{dt} = \mathbf{f}(\phi) \quad (1)$$

where  $\phi = [T, Y_1, Y_2, \dots, Y_{N_s}]^T$  is the local thermochemical state,  $N_s$  is the number of species and  $\mathbf{f}(\phi) = [\dot{T}, \dot{Y}_1, \dot{Y}_2, \dots, \dot{Y}_{N_s}]^T$  is the chemical source term. Essentially, chemistry is solved by integrating Equation (1) over a computational time step  $\Delta t_{CFD}$ :

$$\phi(t_0 + \Delta t_{CFD}) = \phi(t_0) + \int_{t_0}^{t_0 + \Delta t_{CFD}} \mathbf{f}(\phi(t)) dt \quad (2)$$

The nonlinear initial-value problem given by Equation (2) can be obtained using Newton iteration, where a succession of linear equations is solved whose solution eventually converges to the solution of the nonlinear problem. Correspondingly, to advance the solution over a subinterval ( $\Delta t_{ODE}$ ) of the computational fluid dynamic (CFD) time step  $\Delta t_{CFD}$  from time  $t_n$  to  $t_{n+1}$ , the problem can be linearised as follows by neglecting higher-order terms of  $\mathcal{O}(\Delta t_{ODE}^2)$ :

$$(\mathbf{I} - \alpha \mathcal{J}_n \Delta t_{ODE})(\phi_{n+1} - \phi_n) = \mathbf{G}(\Delta t_{ODE}, \mathbf{f}_n) \quad (3)$$

where  $\alpha$  and  $\mathbf{G}(\Delta t_{ODE}, \mathbf{f})$  are respectively the model coefficient and modified source term. Solving Equation (3) from Equation (2) requires an efficient stiff ODE solving framework comprising three major components: (1) ODE-integration algorithm, (2) Jacobian evaluation formulation, and (3) linear system solver. Subsequently, the efficient numerical framework proposed in the present work will be briefly summarised in the three aforementioned aspects.

### 2.2. Time-integration algorithms

Three different ODE-integration algorithms are considered: an implicit Runge–Kutta method (Radau-IIA), an extrapolation method (Seulex), and the BDF method, wherein

the Radau-IIA method will serve as the main workhorse while the other two are used for comparison. The general form of  $s$ -stage Runge–Kutta method reads:

$$\mathbf{G}_i = \mathbf{f} \left( \phi_n + \Delta t_{ODE} \sum_{j=1}^s a_{ij} \mathbf{G}_j \right), i = 1, \dots, s \quad (4)$$

$$\phi_{n+1} = \phi_n + \Delta t_{ODE} \sum_{i=1}^s b_i \mathbf{f}(\mathbf{G}_i(t_n + c_i \Delta t_{ODE})) \quad (5)$$

in which  $a_{ij}$  and  $c_i$  are model coefficients [17]. In the present work, the 3-stage fully implicit Runge–Kutta method, namely Radau-IIA, is employed. This method was first established by Ehle [32] based on the Radau quadrature [17]. As an implicit method, Radau-IIA is  $L$ -stable and of relatively high order (with a fixed order of 5), which means that this method could handle problems with extreme stiffness. Moreover, a method of this high order indicates that large integration steps can be taken for stringent error tolerance conditions. An automatic time step size control strategy is adopted using a combination of theoretical and heuristic relations to achieve the prescribed error tolerance.

To evaluate the numerical accuracy of the implemented Radau-IIA method, we resort to the BDF method as in the CVODE package [33]. In the BDF method, chemistry integration is linearised as a multi-step relation:

$$\sum_{i=0}^j \alpha_i \phi_{n+1-i} = \Delta t_{ODE} \beta \mathbf{f}(\phi_{n+1}) \quad (6)$$

where  $\phi_{n+1-i}$  are the known state vectors evaluated at the past integration steps and  $j$  is the number of previous steps included to construct the current approximation, which determines the accuracy order  $p$ .  $\phi_{n+1}$  is the state vector to be evaluated at the current integration, and  $\alpha_i$  and  $\beta$  are the coefficients of BDF method [34,35]. BDF method is  $A$ -stable only for  $p \leq 2$ , and  $A(\alpha)$  stable for  $p \leq 6$  [17]. For this reason, the BDF method usually takes a maximum order no larger than 6 (typically 5). Furthermore, in the BDF method, a low order approximation with very small  $\Delta t_{ODE}$  is used at the beginning of each  $\Delta t_{CFD}$  and then builds its highest order of accuracy. Such character leads to significant cost concentration at the beginning of each integration, which causes considerable efficiency deterioration when the CFD time step  $\Delta t_{CFD}$  is relatively small. Furthermore, the BDF method in CVODE package uses the direct solvers for the linear systems, which requires more accurate Jacobian formulation than the iterative preconditioned methods [14]. This hinders its computational speedup potential under approximated Jacobian formulations, wherein more internal iterations are needed for the integration based on less accurate Jacobians.

As the BDF method is introduced by coupling with external CVODE package, hence the computational efficiency comparison between Radau-IIA and BDF methods is not straightforward. Hence, the Seulex method shipped with the OpenFOAM standard ODE solver library is taken as the benchmark mainly for efficiency assessment. The Seulex method is an extrapolation-based algorithm that uses a sequence of lower-order solutions to project high-order approximation of  $\phi(t_0 + \Delta t_{ODE})$ . This method splits the integration step  $\Delta t_{ODE}$  into several sub-intervals  $h_k = \Delta t_{ODE}/n_k$ , where  $n_k$  is defined by a sequence [17]:  $n_1 < n_2 < n_3 < \dots < n_k$  (i.e.: 2, 3, 4, 6, 8, 12, ...). Afterward, in each sub-interval, a

first-order semi-implicit Euler method is used to solve Equation (3) using the sub-interval size  $h_k$ . The low-order solution is then used to successively build higher-order approximations via the Aitken-Neville Algorithm [15,17]. As Imren and Haworth suggested that the Seulex method offers significant advantages in accuracy and computational efficiency compared to the BDF method [15], it has become the most preferred stiff ODE solver in the OpenFOAM platform nowadays [36].

### 2.3. Jacobian evaluation methods

The Jacobian matrix  $\mathcal{J}$  is filled with partial derivatives of energy and species equations as  $\mathcal{J}_{1,1} = \frac{\partial \dot{T}}{\partial T}$ ,  $\mathcal{J}_{k+1,1} = \frac{\partial \dot{Y}_k}{\partial T}$ ,  $\mathcal{J}_{1,j+1} = \frac{\partial \dot{T}}{\partial Y_j}$  and  $\mathcal{J}_{k+1,j+1} = \frac{\partial \dot{Y}_k}{\partial Y_j}$ , where  $k = 0, \dots, N_s$  and  $j = 0, \dots, N_s$ . The first three entries are dense, while the remaining part of the Jacobian matrix  $\mathcal{J}_{k+1,j+1}$  contains the species mass fraction temporal derivatives with respect to species mass fractions. As a matter of fact, sparsity of the Jacobian formulation descends from the sparsity of the stoichiometric coefficients matrix  $\nu'$  and  $\nu''$ . However, when third-body and pressure-dependent reactions are considered, the involvement of more colliding partners leads to dense lines in the Jacobian matrix for each species participating in such complex reactions.

To improve the computational efficiency, two main approximations are assumed in the present work. First, since the analytical Jacobian formulation in pyJac [29] is mass fraction based, it is typically dense because all thermo-chemical variables are coupled through the following relation.

$$\mathcal{J}_{k+1,j+1} = \frac{W_k}{\rho} \left( \frac{\partial \dot{\omega}_k}{\partial Y_j} - \frac{\dot{\omega}_k}{\rho} \frac{\partial \rho}{\partial Y_j} \right) \quad (7)$$

wherein  $\rho$  denotes the density of the mixture, while  $W_k$  and  $\dot{\omega}_k$  are the molecular weight and overall production rate of the  $k$ th species. Correspondingly, the approximation was made by keeping  $\rho$  fixed to increase the sparsity of the Jacobian. Recently, Imren [36] has conducted a comparative study on this approximation with the molar and molar concentration based Jacobians, in which comparable solution accuracy was achieved by these three approaches.

Second, because species can be coupled through third body in pressure-dependent reactions, any species involved as a reactant or product in a third-body reaction has a dense row for chemical Jacobian. Since the number of species having enhanced third-body molecularity coefficients is usually limited, the approximation proposed by Schwar et al. [31] was employed. In this approach, a more efficient storage of the molecularity coefficient  $\beta_{j,k} = 1 - \alpha_{j,k}$  was devised. Thus, only the nonzero entries of  $\beta_{j,k}$  corresponding to the species, which has an enhancing behaviour toward modifying the average collision frequency, need to be stored and considered in computing the effective molecularity values. Subsequently, the row in Jacobian matrix related to the species, which participates in third-party reactions are almost completely removed, and only the columns containing derivatives with respect to species with enhanced molecularity coefficients remain nonzero. This approximation results in a much sparser matrix layout and maximises the computational efficiency of the analytical Jacobian.

Although, Equation (3) ensures that the accuracy of  $\mathcal{J}$  does not influence the accuracy of the converged solution, the convergence rate of Newton iterations is dictated by the accuracy of Jacobian and an over-simplified  $\mathcal{J}$  may cause failures in convergence. The

Newton iteration would encounter step rejection and the ODE integrator may use a reduced internal integration step, which adversely affects the total efficiency of the ODE solvers. Additionally, providing more accurate  $\mathcal{J}$  is also favourable for time-integration algorithms to reduce the frequency of evaluating  $\mathcal{J}$ . In Morev et al.'s recent work [20], they also found that the native AAJ formulation fails to deliver a fast solution to stiff ODE with tight ODE convergence tolerances. In this regard, a full analytical Jacobian (FAJ) formulation introduced by Niemeyer et al. [29] as pyJac library is utilised for comparison in the present work. With both FAJ and AAJ formulations, we will demonstrate in the following sections that the accuracy of  $\mathcal{J}$  requires a judicious balance between the convergence of Newton iteration, the Jacobian sparsity as well as linear equation solving.

#### 2.4. Linear system solvers

Each iteration requires solving a linear system, hence linear algebra can dominate the computational cost if unsuitable methods are employed. In the original implementation of OpenFOAM, the system of linear equations is solved by the native Gaussian elimination method with LU decomposition and back substitution (referred to as native linear solver, NLS), which is found to be inefficient for solving dense matrix of small-scale chemistry [20]. Correspondingly, we replaced the native LU decomposition and back substitution operations with more robust subroutines in the LAPACK library [37].

However, as the size of the chemical mechanisms increases, the inherent sparsity of the combustion chemistry suggests that a sparse matrix representation should be preferred. As a result, for large-scale chemical kinetics, sparse matrix representation with associated sparse matrix algebra is implemented in the present framework. Following Imren and Haworth [15], the advanced sparse matrix algebra is realised by the latest version of the KLU library which is a part of the SuiteSparse package [38]. It should be noted that for high fidelity simulation with detailed chemistries as we are concerned, the involved chemical mechanisms may span a wide range in the number of species and reactions. As a matter of fact, the question here is to determine suitable combinations of ODE-integration algorithm, Jacobian evaluation formulation, and linear system solver for a certain level of chemistry complexity, which is one of the main concerns for the present study.

The details of code implementation covering the three aforementioned aspects are summarised in Appendix 1. In the present work, all source codes are compiled using GCC-4.85 compiler with optimisation flag-O3 enabled and all simulations were executed on Intel Xeon X5670 2.93 GHz CPU. Besides, to preclude the influence of load imbalance due to parallel computing, all simulations were performed using a single core. To eliminate any variability caused by cache warmup or other processes, five repeated calculations will be performed to obtain the average execution time when comparing the computational efficiency.

### 3. Results and discussion

#### 3.1. Homogeneous ignition problem

The performance of the new ODE solvers is firstly investigated in the constant-pressure adiabatic ignition problem. All simulations are performed with the chemFOAM solver. Three major aspects, namely the ODE-integration algorithm, Jacobian evaluation formulation, and linear system solver are comprehensively evaluated.



Table 1. Simulation condition and numerical configuration of ethylene-air auto-ignition problem.

Initial temperature [K]	1200
Pressure [atm]	1.0
Equivalence ratio $\phi$	1.0 for C <sub>2</sub> H <sub>4</sub> and Air
CFD time step ( $\Delta t_{CFD}$ ) [s]	$10^{-6}$
End time ( $t_{end}$ ) [s]	$10^{-3}$
Absolute error tolerance ( $\varepsilon_a$ )	$\varepsilon_a^1 = 10^{-14}$ , $\varepsilon_a^2 = 10^{-16}$
Relative error tolerance ( $\varepsilon_r$ )	$10^{-3}$ – $10^{-8}$
Linear system solver	NLS
Jacobian evaluation method	AAJ

### 3.1.1. Integration algorithm

Auto-ignition simulations of the stoichiometric ethylene-air mixture are performed using the 57-species, 269-reactions UCSD mechanism [39] with an initial temperature of 1200 K under atmospheric pressure. Most ODE solvers require the specification of two convergence criteria, i.e. an absolute tolerance ( $\varepsilon_a$ ) and a relative tolerance ( $\varepsilon_r$ ), to determine when the nonlinear problem has been solved to sufficient accuracy. However, because of the differences in the order of accuracy of the ODE solvers, the usage of the tolerances internally in each solver and the accumulation in round-off errors, it cannot assume that all solvers will yield a solution of the same global accuracy under the same convergence criteria. Therefore, two absolute tolerance values  $\varepsilon_a^1 = 10^{-14}$  and  $\varepsilon_a^2 = 10^{-16}$  combined with 7 relative tolerance values  $\varepsilon_r$  ranging from  $10^{-3}$  to  $10^{-8}$  are considered.

The auto-ignition simulations for the ethylene-air mixture was performed with a fixed CFD time interval  $\Delta t_{CFD} = 10^{-6}s$ , which corresponds to a time scale relevant to reactive flow simulation under engine-relevant conditions, and the total simulation time was set at 1.0 ms. The current comparisons focus more on assessing the ODE-integration algorithms, hence the OpenFOAM's native linear system solver (NLS) and approximated Jacobian evaluation method (AAJ) were employed. The corresponding simulation condition and numerical configuration can be found in Table 1.

For quantitative comparison, the numerical accuracy has been defined as the average of relative differences between the calculated results and the reference values over the whole integration interval and for all the cases.

$$E_R(\varphi_i) = \frac{1}{N_{cases}} \sum_{j=1}^{N_{cases}} \frac{1}{t_{end}} \int_{t=0}^{t_{end}} \left| \frac{\varphi_i - \varphi_i^{ref}}{\varphi_i^{ref}} \right| d\tau \quad (8)$$

where  $\varphi_i$  is the  $i$ -th component of the evaluated state vector and  $\varphi_i^{ref}$  is that obtained with very tight tolerances ( $\varepsilon_a = 10^{-20}$ ,  $\varepsilon_r = 10^{-10}$ ) for the corresponding ODE-integration algorithms.

From Figure 1, a general trend is shown that for a larger  $\varepsilon_a$ , the numerical accuracy is less sensitive to  $\varepsilon_r$ , whereas using tighter  $\varepsilon_a$  incurring more computational cost. Regarding the computational accuracy, with a larger  $\varepsilon_a$ , the  $\varepsilon_r$  has little influence on the global prediction accuracy. However, with a tighter  $\varepsilon_a$ , influence of the  $\varepsilon_r$  is slightly different for each ODE algorithm. The Seulex and Radau-IIA methods level off at large relative tolerance on the global prediction accuracy while BDF method shows a more gradual change in accuracy with relative tolerance. This suggests that an appropriate combination of  $\varepsilon_a$  and  $\varepsilon_r$  should be used to control the solution accuracy to the desired level. Figure 1(a)

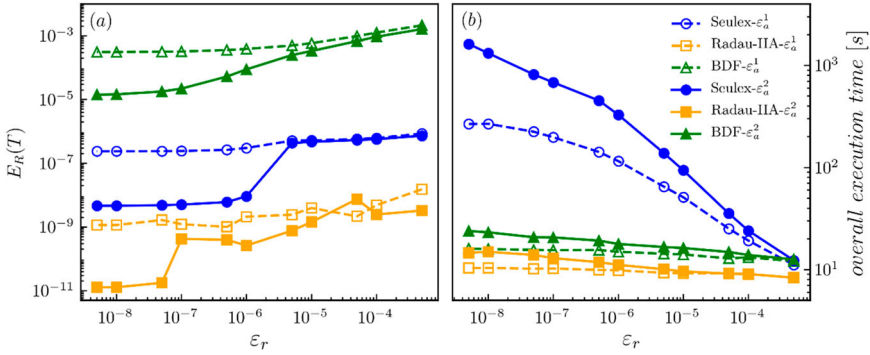


Figure 1. Influence of relative error tolerance  $\varepsilon_r$  on (a) the accuracy in temperature prediction and (b) overall execution time for different ODE-integration algorithms in auto-ignition problem with  $\Delta t_{CFD} = 10^{-6}$ s for physical calculation time  $10^{-3}$ s.

also shows that for the same tolerance condition, the Radau-IIA method obtains the best numerical accuracy. Regarding the computational efficiency, Radau-IIA again obtains the best cost-accuracy trade-off among all three methods. Specifically, Figure 1(b) indicates that the Radau-IIA method achieves this high accuracy while keeping the computational overhead to a minimum level, obtaining orders of computational saving in comparison with the Seulex method. When compared to the BDF method, the computational saving by Radau-IIA is nearly two-fold which is slightly lower.

Furthermore, it can be found from Figure 2 that the order of accuracy of the ODE-integration algorithms is the most influential factor in the cost-accuracy trade-off. Under tight tolerance conditions, the higher order integration algorithms perform better as they are allowed to take much-extended integration steps with comparable accuracy to that of low-order methods. In the meantime, the larger integration step size further leads to fewer function calls including both Jacobian evaluation and linear system solving. As such, the BDF method suffers more from its low order start-up at each new CFD time step causing a deficiency compared to the other two methods. Figure 2(b) presents the mean number of Jacobian evaluations required by different ODE-integration algorithms within each integration step, among which the Radau-IIA requires the least number of Jacobian evaluations. Comparatively, although Seulex is allowed to take a much higher order of accuracy (12 in this work), the cost of evaluating Jacobians would impair its computational efficiency.

As discussed earlier, the re-initialisation problem is a key issue when stiff ODE solver is used as part of an operator-splitting strategy in combustion simulations. To this end, the re-initialisation problem is demonstrated herein with 5 levels of reacting flow integration interval  $\Delta t_{CFD}$  ranging from  $10^{-6}$  to  $10^{-3}$ s with the total integration time  $t_{end}$  being  $10^{-3}$ s. The operating conditions are the same as that in Table 1 except that the error tolerance conditions are fixed to be  $\varepsilon_a = 10^{-14}$  and  $\varepsilon_r = 10^{-4}$ .

From the comparison in Figure 3(a), it is clear that with increasing the computational time interval  $\Delta t_{CFD}$ , Radau-IIA becomes more efficient than the other two methods. Figure 3(b) further confirms that this superiority of Radau-IIA is in some sense owing to the reduced number of internal iterations by taking a larger integration step size. This phenomenon is very similar to that reported by Stone [16] for the comparison of IRK and BDF methods which would be attributed to the high order of the Radau-IIA method at the

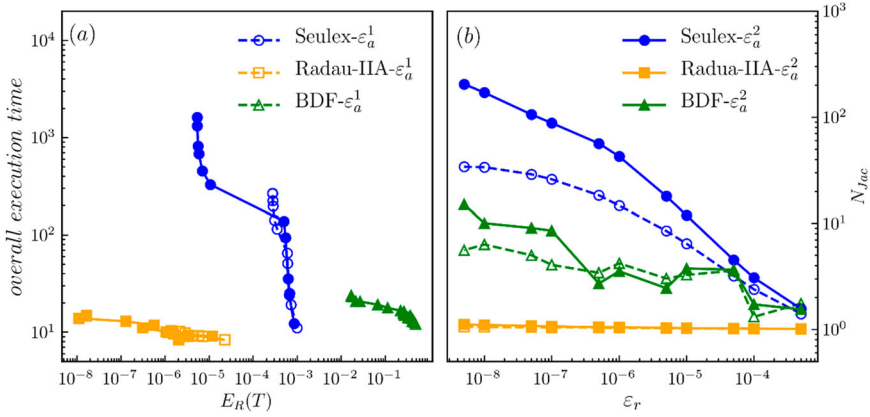


Figure 2. (a) The overall execution time under two levels of  $\varepsilon_a$  versus the mean normalised error of computed temperature result; (b) Influence of relative error tolerance  $\varepsilon_r$  on the mean number of Jacobian evaluations for each CFD time step ( $N_{Jac}$ ) for different ODE-integration algorithms in auto-ignition problem simulated with  $\Delta t_{CFD} = 10^{-6}$ s for physical calculation time  $10^{-3}$ s.

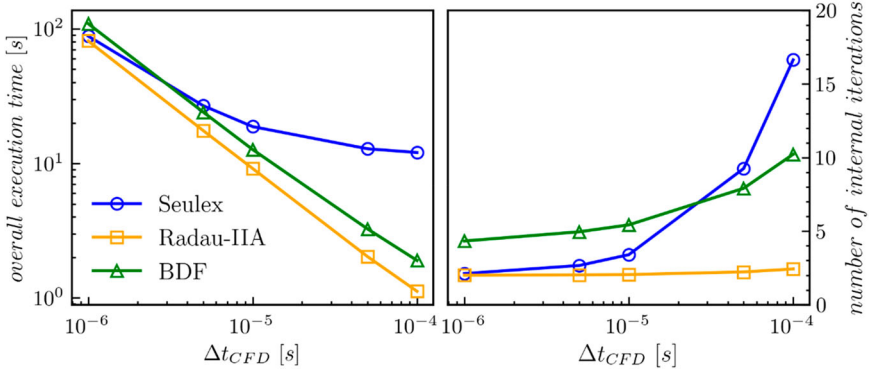


Figure 3. Influence of the  $\Delta t_{CFD}$  on (a) the overall execution time for  $10^{-3}$ s physical simulation time, (b) the number of sub-cycle internal iterations within each computational time interval  $\Delta t_{CFD}$  for different ODE-integration algorithms in auto-ignition problem calculated using  $\varepsilon_a = 10^{-14}$  and  $\varepsilon_r = 10^{-4}$ .

start of each integration interval. This advantage shrinks if the size of integration interval ( $\Delta t_{CFD}$ ) is reduced. As can be seen in Figure 3(a), no obvious difference is observed for all the three ODE-integration algorithms when  $\Delta t_{CFD} < 10^{-6}$  s.

Chemical mechanisms that range in size from 10 to 1389 species as summarised in Table 2 are exploited to assess the performance of ODE-integration algorithms in handling problems with various scales. Furthermore, to cover more thermochemical states, 18 initial value problems (IVPs) are considered for each mechanism, involving two pressure conditions  $p = 1$  and 5atm, three initial temperatures  $T_0 = 800, 1000$  and 1200K, and three initial mixture equivalence ratios  $\phi = 0.5, 1.0$  and 1.5. Each case was integrated for a total time of 3 ms containing the ignition instance with fixed  $\Delta t_{CFD} = 10^{-5}$  and  $10^{-6}$ s respectively, and error tolerance conditions  $\varepsilon_a = 10^{-14}$  and  $\varepsilon_r = 10^{-4}$ . The native linear system solver (NLS) and approximated Jacobian formulation (AAJ) were employed. The corresponding calculation conditions are given in Table 3.

Table 2. Chemical mechanisms used in the present work.

Mechanism	Fuel	# Species	# Reactions
Detailed H <sub>2</sub> [40]	H <sub>2</sub>	10	21
Detailed H <sub>2</sub> -CO [41]	H <sub>2</sub> :CO = 1.0:1.0 in mole fraction	21	62
Skeletal C <sub>12</sub> H <sub>26</sub> [42]	C <sub>12</sub> H <sub>26</sub>	54	269
Skeletal C <sub>7</sub> H <sub>16</sub> [43]	C <sub>7</sub> H <sub>16</sub>	126	680
Skeletal C <sub>12</sub> H <sub>26</sub> [44]	C <sub>12</sub> H <sub>26</sub>	255	1509
Detailed <i>n</i> -heptane [45]	C <sub>7</sub> H <sub>16</sub>	561	2539
Detailed PRF [46]	C <sub>7</sub> H <sub>16</sub>	654	2827
Detail Gasoline [47]	Gasoline surrogate	1389	5935

Table 3. Calculation conditions for chemistry mechanism size sweep.

Initial temperature [K]	800, 1000, 1200
Pressure [atm]	1.0, 5.0
Equivalence ratio $\phi$	0.5, 1.0, 1.5
CFD time step ( $\Delta t_{CFD}$ ) [s]	$10^{-5}, 10^{-6}$
End time ( $t_{end}$ ) [s]	$3 \times 10^{-3}$
Absolute error tolerance ( $\varepsilon_a$ )	$10^{-14}$
Relative error tolerance ( $\varepsilon_r$ )	$10^{-4}$
Linear system solver	NLS
Jacobian evaluation method	AAJ

To evaluate the computational efficiency, the speedup factor  $\chi$  is used to quantify the computational acceleration, which is defined by the ratio of the execution time of one integration algorithm over that of the Seulex method:

$$\chi = \frac{\tau}{\tau_{Seulex}} \quad (9)$$

As is illustrated in Figure 4(a), the Radau-IIA method maintains best efficiency across all mechanisms which is further pronounced for more complex chemistries with a large number of species. Figure 4(b) also confirms the assertion that large  $\Delta t_{CFD}$  is favourable for the Radau-IIA method. Nearly tenfold efficiency improvement over the Seulex method is achieved by the Radau-IIA with large  $\Delta t_{CFD}$ . As a comparison, the BDF method performs poorly when mechanism size scales up. All these degenerated performances are characterised by a surge in the number of integration steps. However, this situation was found to be implicitly caused by the approximation in Jacobian evaluations, which will be further discussed in Section 3.1.2.

Figure 5 shows the profile of the computation budget for various time-integration algorithms over different mechanisms with  $\Delta t_{CFD} = 10^{-6}$  s. Two major time-consuming operations are considered: the LU factorisation (decomposition and back substitution) and Jacobian evaluation. The comparisons suggest that LU factorisation contributes to a dominant expense in large-scale mechanisms. For mechanisms of moderate size ( $100 \leq N_s \leq 500$ ), the cost of the Jacobian evaluation is also non-negligible. These observations emphasise the significance of improving the LU factorisation and Jacobian evaluation subroutines, which will be further addressed in the following sections.

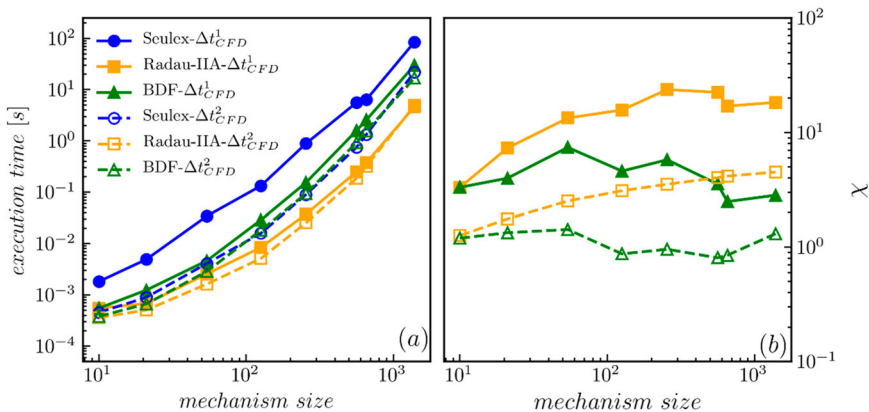


Figure 4. Scaling of (a) the execution time per CFD time step  $\Delta t_{CFD}$ ; (b) the speedup factor for auto-ignition problem calculations with  $\Delta t_{CFD}^1 = 10^{-5}$  and  $\Delta t_{CFD}^2 = 10^{-6}$ s.

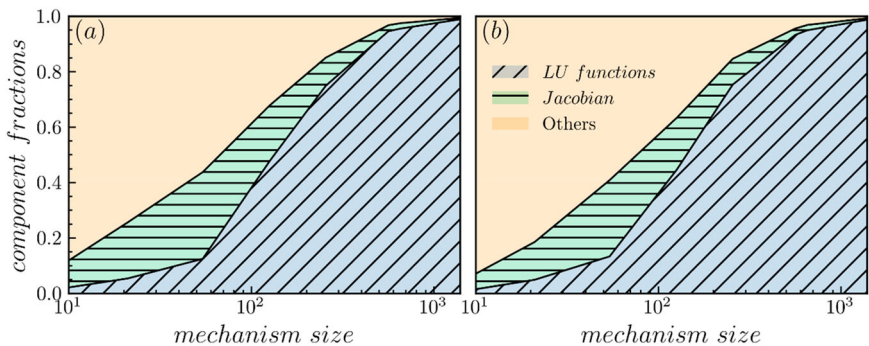


Figure 5. Profiles of computation budget with different mechanisms in auto-ignition calculations using (a) Seulex and (b) Radau-IIA in the simulation of auto-ignition problems with  $\Delta t_{CFD} = 10^{-6}$ s.

### 3.1.2. Jacobian evaluation methods

The full and approximated analytical Jacobian formulations discussed in section 2.3 are systematically assessed based on the configurations in Table 3. It is worth noting that to isolate the influence of the Jacobian evaluation method, the OpenFOAM's native linear system solver (NLS) is employed. The  $\Delta t_{CFD}$  used in these calculations was set to  $10^{-6}$  s.

As illustrated in Section 2.3, the effect of Jacobian evaluation methods on the overall efficiency of the stiff ODE solver is complicated and varies with ODE-integration algorithms and complexity of the chemistry. When the chemical mechanism size is small ( $N_s \leq 100$ ),  $\mathcal{J}$  is essentially dense, thus the extra sparsity obtained with the AAJ method is marginal, whereas the inaccuracy would be detrimental to the convergence of Newton iterations. As can be observed in Figure 6(a), a boost in efficiency is observed for all three integration algorithms if FAJ formulation is provided. However, this trend gradually terminates when the size of the mechanism becomes larger, as it will be more expensive to construct Jacobians to a theoretically accurate extent. This can be found by comparing Figure 6(b,c) that the fraction of Jacobian evaluation expands for moderate-to-large scale mechanisms if FAJ formulation is used.

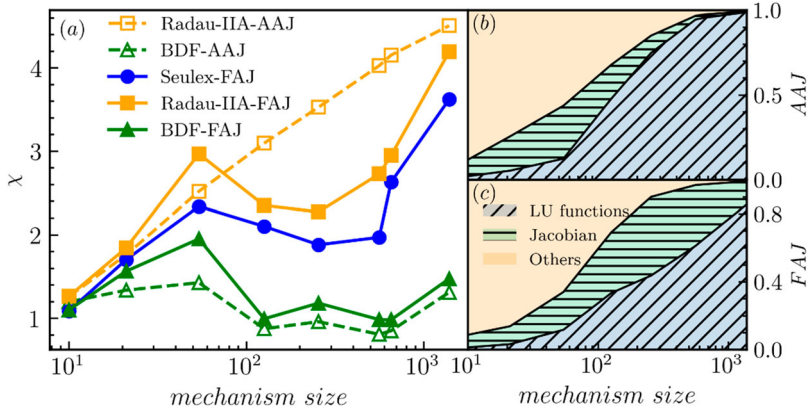


Figure 6. (a) Comparison of speedup factor versus the mechanism size with FAJ and AAJ formulations. The execution time profiling of the Seulex ODE-integration algorithm with (b) AAJ and (c) FAJ in the simulations of auto-ignition problem with  $\Delta t_{CFD} = 10^{-6}$  s.

Additionally, different ODE-integration algorithms suffer differently from the less-accurate Jacobian obtained by AAJ. Clearly, less efficiency attenuation is observed for the Radau-IIA method with AAJ as compared to FAJ formulation for small-scale chemical mechanisms, whereas this tendency is inverses for large mechanisms. For the Seulex method, the efficiency of using FAJ is always superior to that of using AAJ. For the BDF method, it achieves computational speedup up to two folds for mechanism size smaller than 50, while the computational efficiency deteriorates for relatively larger size mechanism.

Figure 7 reveals how the accuracy of Jacobian would affect different integration algorithms in variant size of mechanisms. It worth noting that all three ODE algorithms display similar variation with the increasing mechanism size. Therefore, to make the figure concise, only the results of Seulex and Radau-IIA methods are present. Distinctions are observed not only for the cost of the Jacobian evaluation, but also for solving linear equations. Specifically, Figure 7(b,c) indicate that by using FAJ, integration algorithms require more efforts in large-scale mechanisms to evaluate Jacobian but less effort in solving the linear equation systems. These observations confirm the discussions in Section 2.4 and Figure 6(a) that the effect of Jacobian evaluation methods is two-fold. Additionally, it is presented in Figure 7(b) that a large gap of evaluation cost is observed for the Seulex and Radau-IIA when using the AAJ method. However, this gap shrinks when the evaluation method is replaced with FAJ. In Figure 7(c), the reduced number of Jacobian updates of Seulex also leads to computational saving in solving the linear equation systems. These observations reveal that the Seulex method which suffers more from the massive cost of integration each step, also benefits the most when the frequency of updating Jacobian is reduced by using more accurate Jacobians. In summary, the optimal Jacobian evaluation method for different integration algorithms varies for different mechanisms with a universal trend that the FAJ is more suitable for simulations with small to medium-sized mechanisms, while the AAJ formulation is more suitable for large-scale chemistries.

### 3.1.3. Linear system solvers

In this section, the simulation conditions are the same as in Table 3, except that the linear system solver is replaced by more advanced linear system solvers LAPACK and KLU.

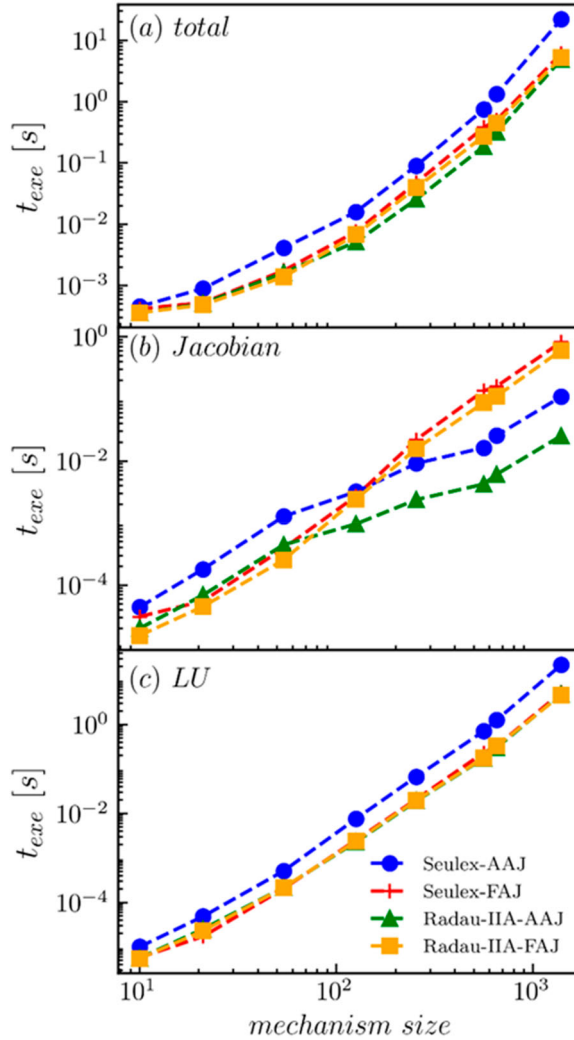


Figure 7. (a) The averaged total execution time, (b) time cost of Jacobian evaluation and (c) time cost of LU function call versus the mechanism size with FAJ and AAJ formulations in simulations of auto-ignition problem with  $\Delta t_{CFD} = 10^{-6}$  s.

To isolate the influence of the linear system solvers, the AAJ formulation for Jacobian evaluation was adopted in this section. It should be noted that the profiling performances are similar for various  $\Delta t_{CFD}$ , thus only the results of  $\Delta t_{CFD} = 10^{-6}$  s is displayed for the sake of concise.

From both Figures 8(a) and 9(a), the improvements are remarkable for both trails using LAPACK and KLU subroutines, as significantly higher speedup factors are obtained for all integration algorithms in large-scale mechanisms. Among all three methods, the Radau-IIA benefits the most by improving the linear system solvers. Nearly 60 times speedup is achieved by LAPACK (Radau-IIA-LAPACK), while a remarkable 200 times speedup is obtained by the KLU case (Radau-IIA-KLU) for simulation of the largest mechanism. However, the Seulex algorithm's overall performance is more complicated with replaced

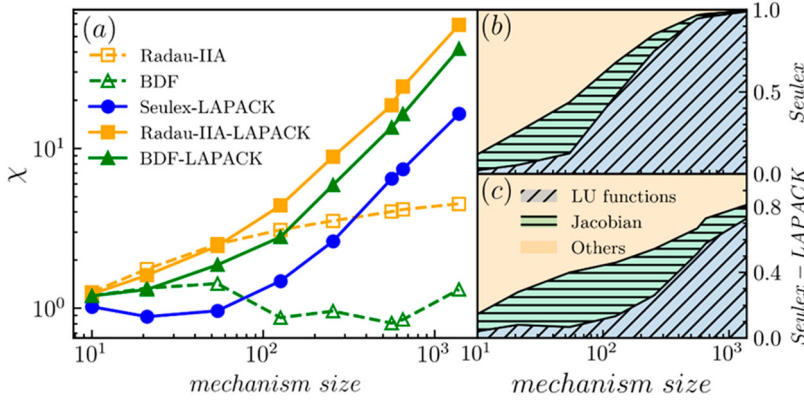


Figure 8. (a) Comparison of speedup factor versus the mechanism size with and without LAPACK linear system solver. The execution time profiling of (b) NLS and (c) LAPACK for the simulations of auto-ignition problem with  $\Delta t_{CFD} = 10^{-6}$  s.

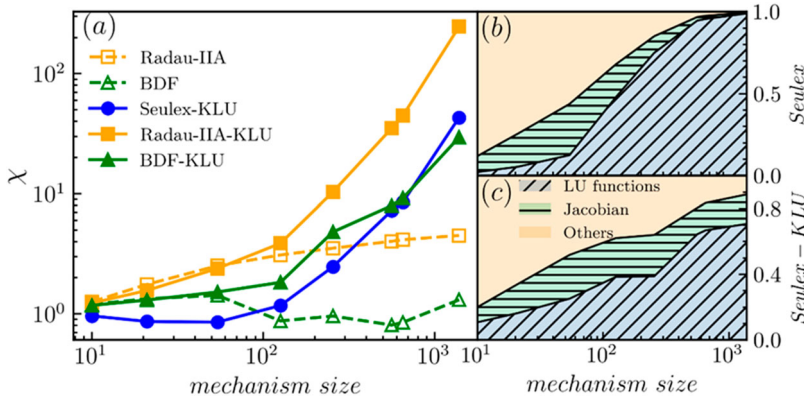


Figure 9. (a) Comparison of speedup factor versus the mechanism size with and without KLU linear system solver. The execution time profiling of (b) NLS and (c) KLU for the simulations of auto-ignition problem with  $\Delta t_{CFD} = 10^{-6}$  s.

linear solvers. It undergoes slight performance degeneration under small size mechanism, while gains significantly computational speedup when the number of species in the mechanism exceeds 100.

Figures 8(b,c) and 9(b,c) show that the computational cost of LU factorisation is significantly reduced for both cases. It is indicated that the LAPACK solver gains more saving in moderate-size mechanisms ( $N_s \sim 100$ ) compared with the sparse KLU solver. To further study the effects of the linear system solvers, Figure 10(a) compares the total execution time obtained by the Seulex integration algorithm with different linear system solvers. For mechanisms of moderate size, the LAPACK solver outperforms the KLU solver, while for large-scale mechanisms, the situation reverses. The execution time profiling of linear system solvers is shown in Figure 10(b), better efficiency is achieved by KLU solver only in large scale cases with  $N_s > 100$ . As the number of species exceeds this threshold, the sparsity of the Jacobian matrix increases to a certain level where the sparse method becomes



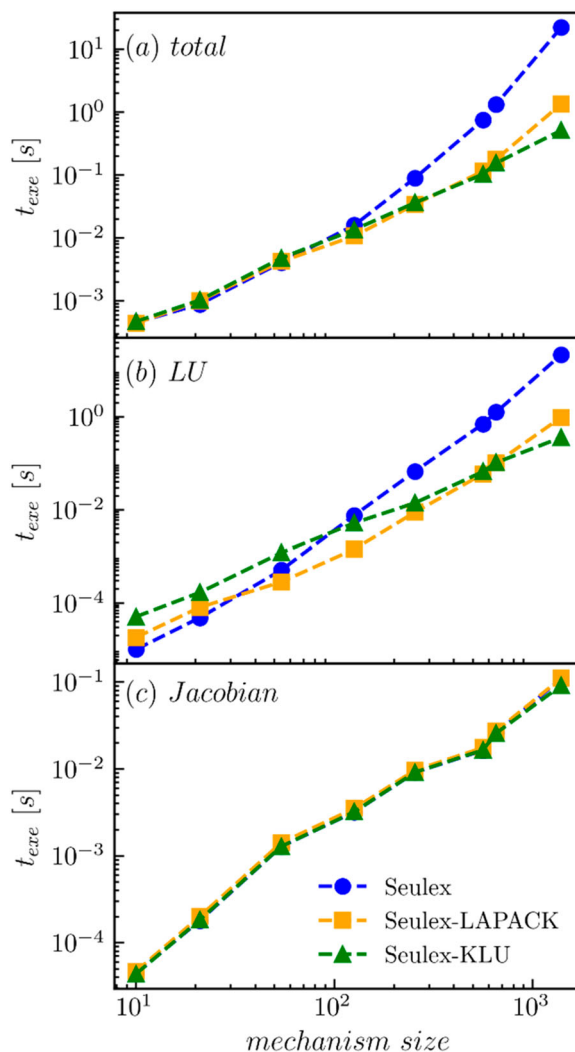


Figure 10. (a) The averaged total execution time, (b) time cost of LU function and (c) time cost of Jacobian evaluation versus the mechanism size obtained by Seulex method with different linear system solvers in the auto-ignition problem simulations.

more effective. Additionally, Figure 10(c) shows that solving method of linear systems should not influence the evaluation of Jacobian.

Further insight into the influence of Jacobian sparsity can be drawn from Figure 11. A graphical representation of the Jacobian matrix sparsity in two zero-dimensional ignition cases, comparing the FAJ and AAJ formulations, is shown in Figure 11. The corresponding Jacobians were evaluated at the instant where the mixture temperature is 400 K above its initial temperature. As can be seen that, for the FAJ formulation, the Jacobian is almost dense since it is mass-fraction based [29]. The matrix sparsity is 46% and 41%, for the  $H_2$ -CO chemical mechanism [41] and  $C_{12}H_{26}$  skeletal mechanism [42], respectively. In contrast, for the AAJ formulation, by ignoring derivatives of density to mass fractions and

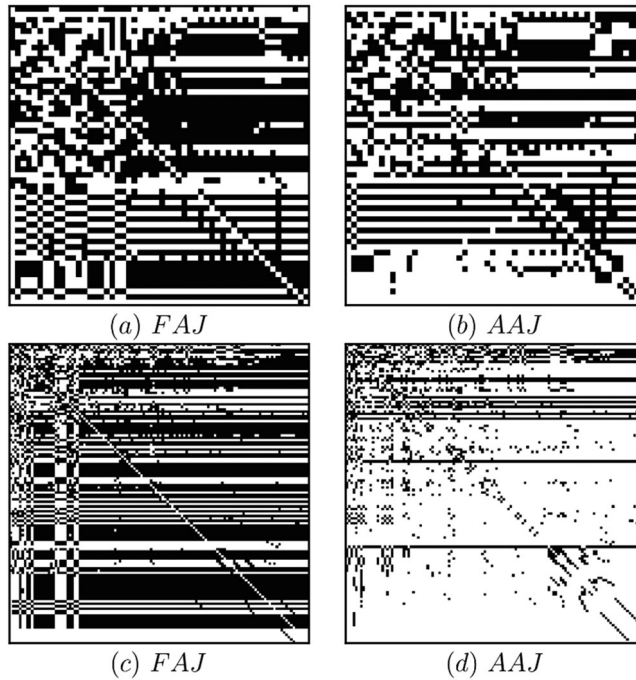


Figure 11. Sparsity pattern of the Jacobian matrix for auto-ignition simulation with (a–b) the  $\text{H}_2\text{-CO}$  chemical mechanism [41] and (c–d) for the  $\text{C}_{12}\text{H}_{26}$  mechanism [42].

invoking a simplified treatment for third-body reactions, the Jacobian sparsity increases to 68% and 57%, respectively. Combining the observations from Figures 10 and 11, it suggests that the cost of constructing Jacobian is less critical than the cost of solving the linear system and the sparsity of the AAJ is more important for efficient solution of the stiff ODE system.

### 3.2. One-dimensional laminar premixed flame

In this section, we intend to study the performance of the integral stiff ODE solving framework in a more realistic condition. For this purpose, a one-dimensional stoichiometric methane–air premixed flame at atmospheric condition was simulated with the reacting-FOAM solver in OpenFOAM. The UCSD mechanism with 57 species and 269 reactions was used, and the initial condition and computational mesh were specified based on the flame profile generated by Cantera [48]. The CFD time step size ( $\Delta t_{CFD}$ ) is constrained with the maximum Curren number 0.2. Based on the converged result obtained by Cantera, CFD simulation was performed for another 1000 time steps with error tolerance condition  $\varepsilon_a = 10^{-14}$  and  $\varepsilon_r = 10^{-4}$ . The numerical accuracy is quantified by the relative error at the last step, with the reference baseline obtained by each individual method using tight tolerance conditions  $\varepsilon_a = 10^{-16}$  and  $\varepsilon_r = 10^{-10}$ .

Figure 12 shows the performance of each integration algorithm (using NLS linear solver and AAJ Jacobian formulation) along the mesh grid. The observations are generally consistent with the auto-ignition calculations. With less number of internal integration steps

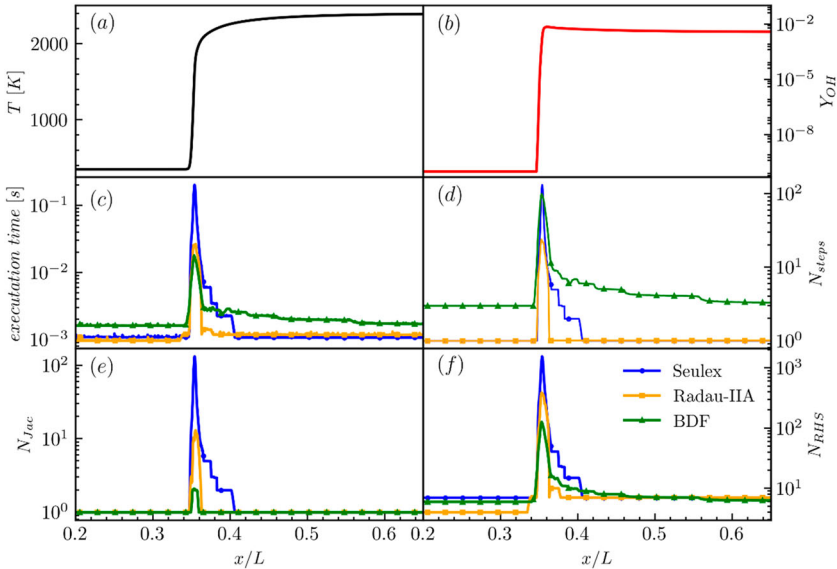


Figure 12. Spatial distribution of (a) temperature, (b) mass fractions of OH, (c) execution time, (d) number of internal integration steps, (e) number of Jacobian evaluations and (f) number of derivative (RHS) evaluations in the simulation of methane–air premixed flame.

as in Figure 12(d) and corresponding function evaluations in Figure 12(e,f) required especially in the vicinity of the flame front, the Radau-IIA method again obtains the best overall efficiency while the BDF method suffers a lot from more integration steps required in the reaction zone. Likewise, the Seulex method needs more number of Jacobian evaluations and an extended spatial range to recover to the state of the non-reactive region.

The overall accuracy of the calculations as the averaged value among all the grid cells is shown in Figure 13. As seen, the Radau-IIA method pursues the same level of relative error compared to the Seulex method. In comparison, a much larger error is observed for BDF method, especially for species in small concentration, i.e. CO and OH, which might be the consequence of the low-order scheme taken by the BDF to restart each new CFD time step.

Correspondingly, the averaged execution time obtained with various integral ODE solvers is listed in Table 4. Clearly, considering both accuracy and calculation efficiency in laminar flame simulation, the configuration of Radau-IIA algorithm associated with the LAPACK solver and FAJ formulation would be preferred, which achieves more than 4.5 times efficiency gain compare to the Seulex algorithm with standard NLS solver and AAJ formulation.

The optimal configuration of stiff ODE solving framework for chemistry with various complexity is further investigated using three typical scales of chemical mechanisms for ethylene (57 species) [39], *n*-heptane (126 species) [43] and *n*-dodecane (255 species) [44]. The computational speedup only considers the time spent for chemistry solving. The results in Figure 14 suggest that, for smaller mechanisms with 57 and 126 species, nearly 2.8–4.6 folds efficiency gain is obtained by the Radau-IIA method with the LAPACK solver and the FAJ formulation compared to OpenFOAM’s native implementation. However, for the mechanisms of large size ( $N_s = 255$ ), it is obvious that the AAJ formulation is more

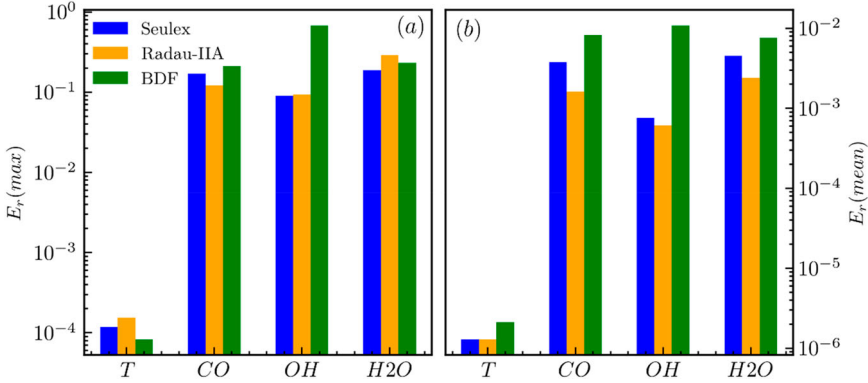


Figure 13. (a) Maximum and (b) averaged relative error in temperature, mass fractions of CO, OH and H<sub>2</sub>O over all mesh points in the simulation of methane–air premixed flame. The reference values were obtained by each individual method using tight tolerance conditions  $\varepsilon_a = 10^{-16}$  and  $\varepsilon_r = 10^{-10}$ .

Table 4. Averaged execution time for chemistry solving of various combinations of ODE-integration algorithm with linear system solvers and Jacobian formulations of methane–air premixed flame simulations.

Integral ODE Solver	AAJ [s]	FAJ [s]
Seulex	10.44	9.38
Seulex-LAPACK	10.26	4.51
Seulex-KLU	12.75	12.33
Radau-IIA	5.75	4.87
Radau-IIA-LAPACK	5.22	2.27
Radau-IIA-KLU	6.31	5.78
BDF	8.22	7.37
BDF-LAPACK	5.95	3.05
BDF-KLU	8.51	6.64

efficient. Additionally, the sparse linear algebra method in the KLU begins overtaking the dense-based LAPACK solver under this mechanism scale.

### 3.3. Three-dimensional turbulent flame simulation

The optimised integral ODE solving framework is further assessed against a well-established laboratory jet flame, Sandia Flame D [49], to demonstrate their performance under realistic turbulent combustion. The Sandia burner has a central fuel pipe of diameter  $D_j = 7.2\text{mm}$  surrounded by a 4.5mm annulus from where a pilot flame is issued. The fuel nozzle injects a methane/air mixture (1:3 by volume) at 49.6 m/s, whereas the pilot stream burns a mixture (equivalence ratio  $\phi = 0.77$ ) with the same nominal equilibrium composition and enthalpy as methane/air mixture, which is issued at a velocity of 11.4 m/s.

The computational domain is a cylinder with a diameter of  $41.67D_j$  and length of  $60D_j$  and is discretised by a stretched grid with 86, 48, and 244 cells in the radial, azimuthal and axial directions (1,042,368 grids in total), respectively. A time-varying boundary condition is applied for the fuel jet velocity to provide realistic turbulence, and the turbulent inflow data is generated by a diffusion-based method [50]. A three-dimensional LES simulation

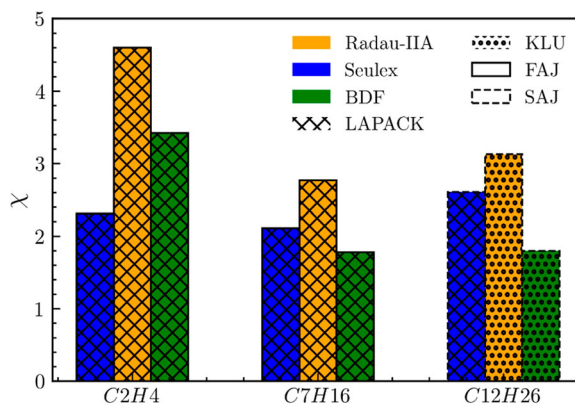


Figure 14. Speedup factors for chemistry solving obtained by the optimal configuration of three integration algorithms in the simulation of 1-D stoichiometric fuel-air premixed flame with different mechanisms: (a) 57 species UCSD mechanism, (b) 126 species n-heptane mechanism and (c) 255 species n-dodecane mechanism, which are detailed in Table 2.

of the flame is performed using reactingFOAM solver with the partially stirred reactor (PaSR) turbulent combustion model. The chemical reaction rates are evaluated based on UCSD detailed chemistry [39]. The LES simulation was first run in parallel to reach a fully developed reactive state, and the performance evaluations for integral ODE solvers were then performed for another 1.0 ms with the  $\Delta t_{CFD}$  being  $10^{-6}$ s in serial to avoid parallel load imbalance. All simulations were conducted with error tolerance condition  $\varepsilon_a = 10^{-14}$  and  $\varepsilon_r = 10^{-4}$ .

For the steady-state flow field, the time-averaged statistics of temperature are collected for 6 flow-through times. Radial profiles at two axial locations are presented in Figure 15. The excellent agreement between the numerical predictions and experimental measurements verified the reliability of physical models and numerical setups for the present simulation, which serves as a prerequisite for the following performance evaluations. From the discussion of 0-D and 1-D scenarios, we can conclude that for small to intermediate-scale chemical mechanisms, dense matrix-based LAPACK linear solver associated with FAJ formulation is preferred. Therefore, we resort to this optimal combination for Flame D simulation while three ODE-integration algorithms are comprehensively assessed.

Figure 16 compares the mean execution time and speedup factors for chemistry solving with three ODE integration algorithms. From Figure 16, Radau-IIA with LAPACK solver and FAJ formulation obtains the best computational efficiency yielding a speedup by a factor of 2.66 compared to the NLS solver and AAJ formulation. Furthermore, with LAPACK linear system solver and FAJ formulation, the Seulex and BDF methods also attain 1.68 and 1.94 folds of computational speedup. This again demonstrated the benefits of using a more sophisticated linear system solver (LAPACK) and Jacobian formulation (FAJ) in small to medium-sized chemical mechanisms.

Specifically, the spatially distributed instantaneous performance of the optimal configuration for each integration algorithm is displayed in Figure 17. From Figure 17(a-c), it is clear that most of the computational costs are concentrated in the vicinity of the reaction zone near the flame front. This situation is attributed to both more numbers of integration steps and Jacobian evaluations in this region. For the first factor, Radau-IIA and Seulex algorithms take almost the same level of steps to integrate one CFD time step, which is

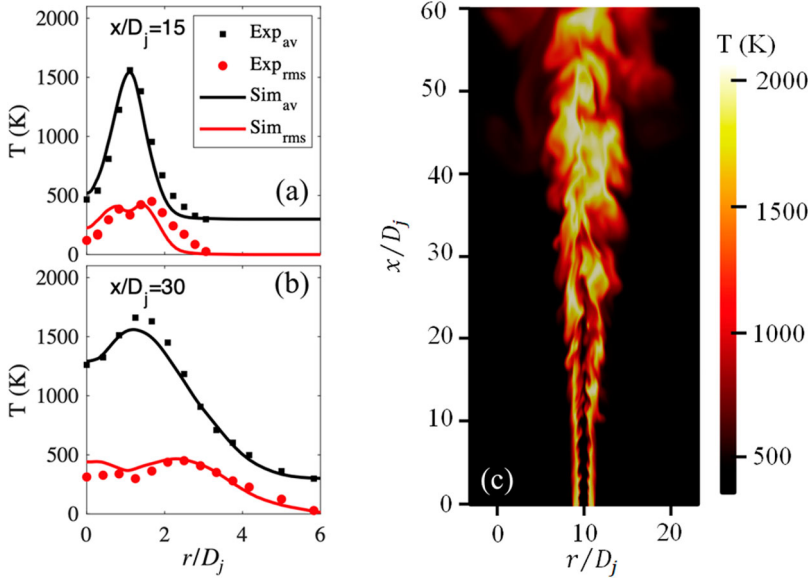


Figure 15. Time averaged radial profiles of temperature at (a)  $x/D_j = 15$ , (b)  $x/D_j = 30$  and (c) instantaneous temperature contour of Sandia flame D.

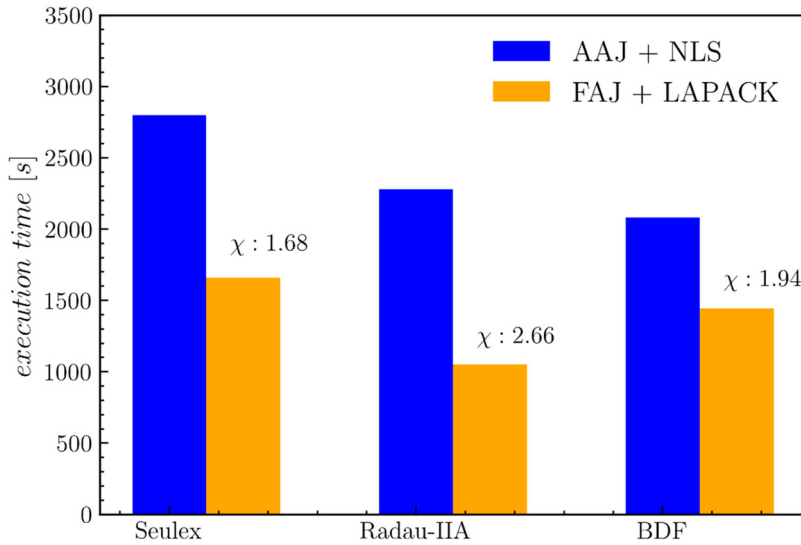


Figure 16. The averaged execution time for chemistry solving of the integral ODE solvers with optimal configuration (FAJ + LAPACK) and their native-implemented competitors (AAJ + NLS) in the simulation of Sandia Flame D.

much smaller than that of the BDF method, indicating that one step method without suffering from the deficient low-order start-up performs better than the multi-step method. At the same time, the frequency of Jacobian evaluations required by Radau-IIA is significantly less than that of the Seulex method, which reduces the computational cost of each integration step and further contributes to the total efficiency gain of this algorithm.

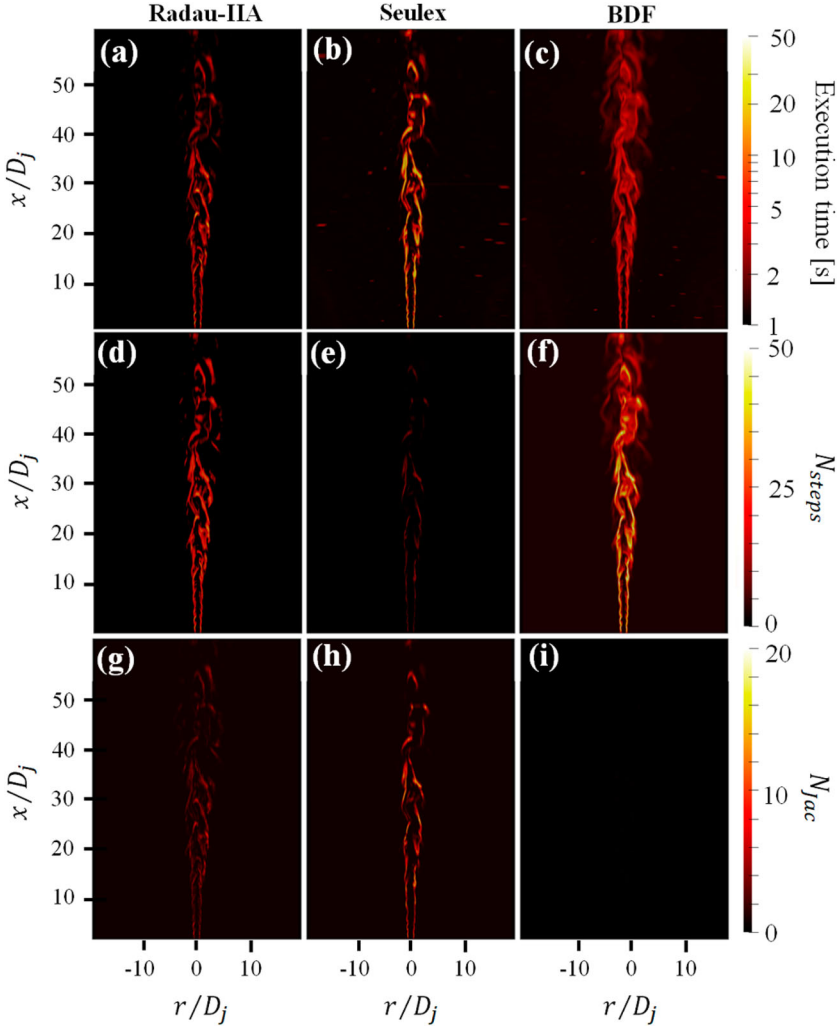


Figure 17. The instantaneous contour of (a)–(c) execution time (d)–(f) number of integrating steps (g)–(i) number of Jacobian evaluations required by integration algorithms using both LAPACK solver and FAJ formulation in the simulation of Sandia Flame D.

#### 4. Concluding remarks

The current work extends the capability of OpenFOAM for accurate yet efficient solution of stiff ODEs involved in combustion simulations of hydrocarbon fuels. This efficient framework consists of three major pillar modules. First, the high-order implicit Runge–Kutta (Radau-IIA) and multi-step backward differentiation formulation (BDF) algorithms were implemented and compared against the native Seulex algorithm. Second, the native approximated analytical Jacobian formulation was replaced by the fully analytical one provided by the open-source pyJac package, which minimises the inaccuracy in Jacobian evaluation. Last, more sophisticated and robust linear system solvers including the dense-based LAPACK and sparse-based KLU packages were coupled with the ODE integrators for further efficiency pursue. Performance and accuracy assessment of the proposed

framework has been conducted for chemical mechanisms ranging in size from 10 to 1389 species, and for three different configurations including the homogeneous reactor, one-dimensional laminar flame, and three-dimensional turbulent combustion.

The homogeneous ignition results suggest that the Radau-IIA method is the most efficient one for large integration intervals when using either FAJ or AAJ formulation for Jacobian evaluation. This superiority of the high-order method can be attributed to low start-up cost, large possible step size and less Jacobian evaluation, whereas the differences between the Radau-IIA, Seulex, and BDF methods decrease as the CFD time step is reduced.

As for the Jacobian evaluation formulation, its influence on the overall trade-off between accuracy and efficiency is two-fold. For small size chemical mechanisms of species about 50–100, the FAJ formulation achieves 2 times computational speedups in calculations with the Seulex ODE-integration algorithm. However, for large-scale mechanisms, the AAJ formulation is more advantageous, as it enhances the sparsity of the chemical Jacobian, which facilitates the subsequent linear system solver. Moreover, among all the three ODE-integration algorithms, Radau-IIA is the most robust one to accommodate AAJ formulation.

The adoption of more advanced linear system solvers is beneficial for improving the efficiency of the ODE solvers, wherein dense-based LAPACK solver is more preferred in small to moderate-scale mechanisms ( $50 < N_s < 500$ ) while sparse-based KLU solver is more suitable for large-scale mechanisms ( $N_s > 500$ ). The Radau-IIA algorithm gains the most computational saving from the KLU solver, obtaining two orders of magnitude speedup in the mechanism with 1389 species.

For practical turbulent combustion simulations with relatively small mechanisms ( $N_s < 200$ ), the combination of the Radau-IIA ODE integration algorithm associated with FAJ Jacobian formulation and LAPACK linear system solver is optimal in accuracy and efficiency trade-off, whereby a 2.66 speedup can be obtained compared to the standard Seulex solver. Moreover, it worth acknowledging that the calculation speedup varies under different operating condition, chemical mechanism and ODE solver configuration. However, the promising results herein indicate that improvements in the ODE solution methodology offer an avenue for more efficient reacting flow simulations.

### Acknowledgements

The code will be available from the corresponding author upon reasonable request.

### Disclosure statement

No potential conflict of interest was reported by the author(s).

### Funding

This work was supported by National Key Project [grant number GJXM92579].

### References

- [1] Q. Yang, P. Zhao, and H. Ge, *ReactingFoam-SCI: An open source CFD platform for reacting flow simulation*. *Comput. Fluids* 190 (2019), pp. 114–127.



- [2] T. Lu and C.K. Law, *Toward accommodating realistic fuel chemistry in large-scale computations*. Prog. Energy Combust. Sci. 35 (2009), pp. 192–215.
- [3] J.H. Chen, *Petascale direct numerical simulation of turbulent combustion – fundamental insights towards predictive models*. Proc. Combust. Inst. 33 (2011), pp. 99–123.
- [4] T. Lu and C.K. Law, *A directed relation graph method for mechanism reduction*. Proc. Combust. Inst. 30 (2005), pp. 1333–1341.
- [5] L. Liang, J.G. Stevens, and J.T. Farrell, *A dynamic adaptive chemistry scheme for reactive flow computations*. Proc. Combust. Inst. 32 (2009), pp. 527–534.
- [6] S.B. Pope, *Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation*. Combust. Theor. Model. 1 (1997), pp. 41–63.
- [7] L. Lu and S.B. Pope, *An improved algorithm for in situ adaptive tabulation*. J. Comput. Phys. 228 (2009), pp. 361–386.
- [8] F. Perini, *High-dimensional, unsupervised cell clustering for computationally efficient engine simulations with detailed combustion chemistry*. Fuel 106 (2013), pp. 344–356.
- [9] D. Zhou, K.L. Tay, H. Li, and W. Yang, *Computational acceleration of multi-dimensional reactive flow modelling using diesel/biodiesel/jet-fuel surrogate mechanisms via a clustered dynamic adaptive chemistry method*. Combust. Flame 196 (2018), pp. 197–209.
- [10] F.E. Hernández Pérez, N. Mukhadiyev, X. Xu, A. Sow, B.J. Lee, R. Sankaran, and H.G. Im, *Direct numerical simulations of reacting flows with detailed chemistry using many-core/GPU acceleration*. Comput. Fluids 173 (2018), pp. 73–79.
- [11] F. Contino, H. Jeanmart, T. Lucchini, and G. D’Errico, *Coupling of in situ adaptive tabulation and dynamic adaptive chemistry: An effective method for solving combustion in engine simulations*. Proc. Combust. Inst. 33 (2011), pp. 3057–3064.
- [12] C.F.H. Curtiss and J. O, *Integration of stiff equations*. Proc. Natl. Acad. Sci. USA 38 (1952), pp. 235–243.
- [13] C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [14] M.J. McNenly, R.A. Whitesides, and D.L. Flowers, *Faster solvers for large kinetic mechanisms using adaptive preconditioners*. Proc. Combust. Inst. 35 (2015), pp. 581–587.
- [15] A. Imren and D.C. Haworth, *On the merits of extrapolation-based stiff ODE solvers for combustion CFD*. Combust. Flame 174 (2016), pp. 1–15.
- [16] C.P. Stone and F. Bisetti, *Comparison of ODE solver for chemical kinetics and reactive CFD applications*, 52nd Aerospace Sciences Meeting, 2014.
- [17] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer, Berlin, 1996.
- [18] N.J. Curtis, K.E. Niemeyer, and C.-J. Sung, *Using SIMD and SIMT vectorization to evaluate sparse chemical kinetic Jacobian matrices and thermochemical source terms*. Combust. Flame 198 (2018), pp. 186–204.
- [19] T. Sauer, *Numerical Analysis*, Pearson, Harlow, 2012.
- [20] I. Morev, B. Tekgül, M. Gadalla, A. Shahanaghi, J. Kannan, S. Karimkashi, O. Kaario, and V. Vuorinen, *Fast reactive flow simulations using analytical Jacobian and dynamic load balancing in OpenFOAM*. Phys. Fluids 34 (2022), pp. 021801.
- [21] F. Perini, E. Galligani, and 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*. Combust. Flame 161 (2014), pp. 1180–1195.
- [22] C. Xu, Y. Gao, Z. Ren, and T. Lu, *A sparse stiff chemistry solver based on dynamic adaptive integration for efficient combustion simulations*. Combust. Flame 172 (2016), pp. 183–193.
- [23] M.A. Hansen and J.C. Sutherland, *On the consistency of state vectors and Jacobian matrices*. Combust. Flame 193 (2018), pp. 257–271.
- [24] S. Eberhardt and S. Imlay, *Diagonal implicit scheme for computing flows with finite rate chemistry*. J. Thermophys. Heat Transfer 6 (1992), pp. 208–216.
- [25] Y. Ju, *Lower-upper scheme for chemically reacting flow with finite rate chemistry*. AIAA J. 33 (1995), pp. 1418–1425.
- [26] B. Savard, Y. Xuan, B. Bobbitt, and G. Blanquart, *A computationally-efficient, semi-implicit, iterative method for the time-integration of reacting flows with stiff chemistry*. J. Comput. Phys. 295 (2015), pp. 740–769.

- [27] C. Safta, H.N. Najm, and O. Knio, *TChem – a software toolkit for the analysis of complex kinetic models*. *Telettraffic Sci Eng.* 1 (2011), pp. 907–916.
- [28] F. Perini, E. Galligani, and R.D. Reitz, *An analytical Jacobian approach to sparse reaction kinetics for computationally efficient combustion modeling with large reaction mechanisms*. *Energy Fuels* 26 (2012), pp. 4804–4822.
- [29] K.E. Niemeyer, N.J. Curtis, and C.-J. Sung, *Pyjac: analytical jacobian generator for chemical kinetics*. *Comput. Phys. Commun.* 215 (2017), pp. 188–203.
- [30] OpenFOAM. Available at: <https://openfoam.org/version/7/>.
- [31] D.A. Schwer, J.E. Tolsma, W.H. Green, and P.I. Barton, *On upgrading the numerics in combustion chemistry codes*. *Combust. Flame* 128 (2002), pp. 270–291.
- [32] B.L. Ehle, *A-stable methods and Padé approximations to the exponential*. *SIAM J. Math. Anal.* 4 (1973), pp. 671–680.
- [33] S.D. Cohen and A.C. Hindmarsh, *CVODE, A stiff/nonstiff ODE solver in C*. *Comput. Phys.* 10 (1996), pp. 138–143.
- [34] G. Byrne and A. Hindmarsh, *A polyalgorithm for the numerical solution of ordinary differential equations*. *ACM Trans. Math. Softw.* 1 (1975), pp. 71–96.
- [35] P.N. Brown, G.D. Byrne, and A.C. Hindmarsh, *VODE: A variable-coefficient ODE solver*. *SIAM J. Sci. Stat. Comput.* 10 (1989), pp. 1038–1051.
- [36] A. Imren, *A detailed error quantification analysis of extrapolation-based stiff ODE solvers for combustion CFD*. *Flow Turbul. Combust.* (2022). <https://doi.org/10.1007/s10494-022-00369-z>.
- [37] E. Anderson, Z. Bai, C. Bischof, S. Blackford, and D. Sorensen, *LAPACK User's Guide*, 3rd ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
- [38] S. Kolodziej, M. Aznaveh, M. Bullock, J. David, and R. Sandstrom, *The SuiteSparse matrix collection website interface*. *J. Open Source Softw.* 4 (2019), pp. 1244.
- [39] *Chemical-Kinetic Mechanisms for Combustion Applications*. San Diego Mechanism Web Page, Mechanical and Aerospace Engineering (Combustion Research), University of California at San Diego.
- [40] M. O'Connaire, H.J. Curran, J.M. Simmie, W.J. Pitz, and C.K. Westbrook, *A comprehensive modeling study of hydrogen oxidation*. *Int. J. Chem. Kinet.* 36 (2004), pp. 603–622.
- [41] A. Kéromnès, W.K. Metcalfe, K.A. Heufer, N. Donohoe, A.K. Das, C.-J. Sung, J. Herzler, C. Naumann, P. Griebel, O. Mathieu, M.C. Krejci, E.L. Petersen, W.J. Pitz, and H.J. Curran, *An experimental and detailed chemical kinetic modeling study of hydrogen and syngas mixture oxidation at elevated pressures*. *Combust. Flame* 160 (2013), pp. 995–1011.
- [42] T. Yao, Y. Pei, B.-J. Zhong, S. Som, T. Lu, and K.H. Luo, *A compact skeletal mechanism for n-dodecane with optimized semi-global low-temperature chemistry for diesel engine simulations*. *Fuel* 191 (2017), pp. 339–349.
- [43] Y. Chen and J.-Y. Chen, *Towards improved automatic chemical kinetic model reduction regarding ignition delays and flame speeds*. *Combust. Flame* 190 (2018), pp. 293–301.
- [44] K. Narayanaswamy, P. Pepiot, and H. Pitsch, *A chemical mechanism for low to high temperature oxidation of n-dodecane as a component of transportation fuel surrogates*. *Combust. Flame* 161 (2014), pp. 866–884.
- [45] T. Lu, C.K. Law, C.S. Yoo, and J.H. Chen, *Dynamic stiffness removal for direct numerical simulations*. *Combust. Flame* 156 (2009), pp. 1542–1551.
- [46] H.J. Curran, P. Gaffuri, W.J. Pitz, and C.K. Westbrook, *A comprehensive modeling study of n-heptane oxidation*. *Combust. Flame* 114 (1998), pp. 149–177.
- [47] M. Mehl, W.J. Pitz, C.K. Westbrook, and H.J. Curran, *Kinetic modeling of gasoline surrogate components and mixtures under engine conditions*. *Proc. Combust. Inst.* 33 (2011), pp. 193–200.
- [48] D.G. Goodwin, H.K. Moffat, and R.L. Speth, *Cantera: An Object-Oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes. Version 2.2.0*, 2015.
- [49] R.S. Barlow and J.H. Frank, *Effects of turbulence on species mass fractions in methane/air jet flames*. *Symp. (Int.) Combust.* 27 (1998), pp. 1087–1095.
- [50] A. Kempf, M. Klein, and J. Janicka, *Efficient generation of initial- and inflow-conditions for transient turbulent flows in arbitrary geometries*. *Flow Turbul. Combust.* 74 (2005), pp. 67–84.
- [51] Intel Math Kernel Library. Available at <https://software.intel.com/enus/intel-mkl>.

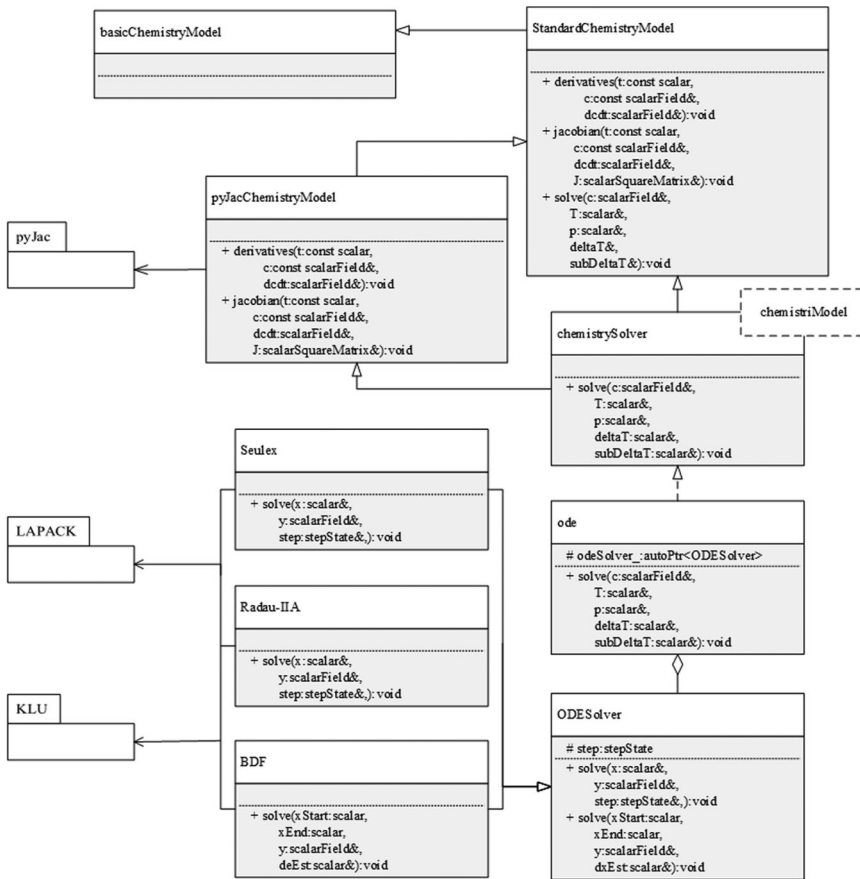


Figure A1. Schematic class diagram of the implementations.

## Appendix 1

The efficient stiff ODE solving framework proposed in the present work is implemented in OpenFOAM 7. As for the ODE-integration algorithms, the Radau-IIA is implemented as a new ODE solver and is registered to the ODESolver. Since the Radau-IIA algorithm involves complex matrix operation, a complex version of LU factorisations is implemented in the linear algebra module. For the BDF method, the CVODE [33] package is linked to OpenFOAM through an interface with ODESolver. The Seulex method is native to the current version of OpenFOAM.

For linear system solvers, the dense matrix-based LAPACK library (Linear Algebra PACKage) [37] is employed for the Matrix blocking operations. The MKL [51] library of version 2018 is selected to provide the basic LAPACK support. Following Imren and Haworth [15], the sparse matrix algebra is realised by the latest version of KLU which is a part of the SuiteSparse package [38]. Since the KLU library requires inputs of matrices in sparse representations (CSC format), a matrix format conversion between OpenFOAM's dense form to KLU's sparse form is implemented.

The approximated analytical Jacobian (AAJ) formulation method is native to OpenFOAM 7 implemented by the standardChemistryModel. The full analytical Jacobian (FAJ) method is implemented by coupling the pyJac library with OpenFOAM. Therefore, a new class, namely pyJacChemistryModel, derived from the parent class, standardChemistryModel, is introduced to override the native function of Jacobian evaluation. The corresponding main class diagram of all the implementations is shown in Figure A1.