A study of direct and Krylov iterative sparse solver techniques to approach linear scaling of the integration of Chemical Kinetics with detailed combustion mechanisms

Federico Perini[a,*], Emanuele Galligani[b], Rolf D. Reitz[a],

[a] *University of Wisconsin–Madison Engine Research Center, 1500 Engineering Drive, Madison, WI 53706, U.S.A.*
[b] *Dipartimento di Ingegneria Enzo Ferrari, Università di Modena e Reggio Emilia, strada Vignolese 905/A, 41125 Modena, Italy*

Abstract

The integration of the stiff ODE systems associated with chemical kinetics is the most computationally demanding task in most practical combustion simulations. The introduction of detailed reaction mechanisms in multi-dimensional simulations is limited by unfavorable scaling of the stiff ODE solution methods with the mechanism's size. In this paper, we compare the efficiency and the appropriateness of direct and Krylov subspace sparse iterative solvers to speed-up the integration of combustion chemistry ODEs, with focus on their incorporation into multi-dimensional CFD codes through operator splitting. A suitable preconditioner formulation was addressed by using a general-purpose incomplete *LU* factorization method for the chemistry Jacobians, and optimizing its parameters using ignition delay simulations for practical fuels. All the calculations were run using a same efficient framework: SpeedCHEM, a recently developed library for gas-mixture kinetics that incorporates a sparse analytical approach for the ODE system functions. The solution was integrated through direct and Krylov subspace iteration implementations with different backward differentiation formula integrators for stiff ODE systems: LSODE, VODE, DASSL. Both ignition delay calculations, involving reaction mechanisms that ranged from 29 to 7171 species, and multi-dimensional internal combustion engine simulations with the KIVA code were used as test cases. All solvers showed similar robustness, and no integration failures were observed when using ILUT-preconditioned

*Corresponding author
Email addresses: perini@wisc.edu (Federico Perini), emanuele.galligani@unimore.it (Emanuele Galligani), reitz@engr.wisc.edu (Rolf D. Reitz)

Krylov enabled integrators. We found that both solver approaches, coupled with efficient function evaluation numerics, were capable of scaling computational time requirements approximately linearly with the number of species. This allows up to three orders of magnitude speed-ups in comparison with the traditional dense solution approach. The direct solvers outperformed Krylov subspace solvers at mechanism sizes smaller than about 1000 species, while the Krylov approach allowed more than 40% speed-up over the direct solver when using the largest reaction mechanism with 7171 species.

Keywords: chemical kinetics, sparse analytical Jacobian, Krylov iterative methods, preconditioner, ILUT, stiff ODE, combustion, SpeedCHEM

---

1. Introduction

The steady increase in computational resources is fostering research of cleaner and more efficient combustion processes, whose target is to reduce dependence on fossil fuels and to increase environmental sustainability of the economies of industrialized countries [1]. Realistic simulations of combustion phenomena have been made possible by thorough characterization and modelling of both the fluid-dynamic and thermal processes that define local transport and mixing in high temperature environments, and of the chemical reaction pathways that lead to fuel oxidation and pollutant formation [2]. Detailed chemical kinetics modelling of complex and multi-component fuels has been achieved through extensive understanding of fundamental hydrocarbon chemistry [3] and through the development of semi-automated tools [4] that identify possible elementary reactions based on modelling of how molecules interact. These detailed reaction models can add thousands of species and reactions. However, often simple phenomenological models are preferred to even skeletal reaction mechanisms that feature a few tens/hundreds of species in "real-world" multidimensional combustion simulations, due to the expensive computational requirements that their integration requires, even on parallel architectures, when using conventional numerics. The stiffness of chemical kinetics ODE systems, caused by the simultaneous co-existence of broad temporal scales, is a further complexity factor, as the time integrators have to advance the solution using extremely small time steps to guarantee reasonable accuracy, or to compute expensive implicit iterations while solving the nonlinear systems of equations [5].

Many extremely effective approaches have been developed in the last few decades to alleviate the computational cost associated with the integration of chemical kinetics ODE systems for combustion applications, targeting: the stiffness of the reactive system; the number of species and reactions; the overall number of calculations; or the main ordinary differential equations (ODE) system dynamics, by identifying main low-dimensional manifolds [6–16]. A thorough review of reduction methods is addressed in [2]. From the point of view of integration numerics, some recent approaches have shown the possibility to reduce computational times by orders of magnitude in comparison with the standard, dense ODE system integration approach, without introducing simplifications into the system of ODEs, but instead improving the computational performance associated with: 1) efficient evaluation of computationally expensive functions, 2) improved numerical ODE solution strategy, 3) architecture-tailored coding.

As thermodynamics-related and reaction kinetics functions involve expensive mathematical functions such as exponentials, logarithms and powers, approaches that make use of data caching, including in-situ adaptive tabulation (ISAT) [17] and optimal-degree interpolation [18], or equation formulations that increase sparsity [19], or approximate mathematical libraries [20], can significantly reduce the computational time for evaluating both the ODE system function and its Jacobian matrix. As far as the numerics internal to the ODE solution strategy, the adoption of sparse matrix algebra in treating the Jacobian matrix associated with the chemistry ODE system is acknowledged to reduce the computational cost of the integration from the order of the cube number of species, $O(n_s^3)$, down to a linear increase, $O(n_s)$ [21–23].

Implicit, backward-differentiation formula methods (BDF) are among the most effective methods for integrating large chemical kinetics problems [24, 25] and packages such as VODE [26], LSODE [27], DASSL [28] are widely adopted. Recent interest is being focused on applying different methods that make use of Krylov subspace approximations to the solution of the linear system involving the problem's Jacobian, such as Krylov-enabled BDF methods [29], Rosenbrock-Krylov methods [30], exponential methods [31]. As far as architecture-tailored coding is concerned, some studies [32, 33] have shown that solving chemistry ODE systems on graphical processing units (GPUs) can significantly reduce the dollar-per-time cost of the computation. However, the potential achieveable per GPU processor is still far from full optimization, due to the extremely sparse memory access that chemical kinetics have in matrix-vector operations, and due to small problem sizes in compari-

son to the available number of processing units. Furthermore, approaches for chemical kinetics that make use of standardized multi-architecture languages such as OpenCL are still not openly available.

Our approach, available in the `SpeedCHEM` code, a recently developed research library written in modern Fortran language, addresses both issues through the development of new methods for the time integration of the chemical kinetics of reactive gaseous mixtures [18]. Efficient numerics are coupled with detailed or skeletal reaction mechanisms of arbitrarily large size, e.g., $n_s \geq 10^4$, to achieve significant speed-ups compared to traditional methods especially for the small-medium mechanism sizes, $100 \leq n_s \leq 500$, which are of major interest to multi-dimensional simulations. In the code, high computational efficiency is achieved by evaluating the functions associated with the chemistry ODE system throughout the adoption of optimal-degree interpolation of expensive thermodynamic functions, internal sparse algebra management of mechanism-related quantities, and sparse analytical formulation of the Jacobian matrix. This approach allows a reduction in CPU times by almost two orders of magnitude in ignition delay calculations using a reaction mechanism with about three thousand species [34], and was capable of reducing the total CPU time of practical internal combustion engine simulations with skeletal reaction mechanisms by almost one order of magnitude in comparison with a traditional, dense-algebra-based reference approach [35, 36].
In this paper, we describe the numerics internal to the ODE system solution procedure. The flexibility of the software framework allows investigations of the optimal solution strategies, by applying different numerical algorithms to a numerically consistent, and computationally equally efficient, problem formulation. Some known effective and robust ODE solvers, as LSODE [27], VODE [26], DASSL [28], used for the combustion chemistry integration [18], are based on backward differentiation formulae (BDF) that include implicit integration steps, requiring repeated linear system solutions associated with the chemistry Jacobian matrix, as part of an iterative Newton procedure. Based on the same computational framework and BDF integration procedure, the effective efficiency of the sparse direct and preconditioned iterative Krylov subspace solvers reported in Table 1 was studied. To consider the effects of reaction mechanism size, a common matrix of reaction mechanisms for typical hydrocarbon fuels and fuel surrogates, spanning from $n_s = 29$ up to $n_s = 7171$ species was used, whose details are reported in Table 1.

The paper is structured as follows. A description of the modelling approach adopted for the simulations is reported, including the major steps of the BDF integration procedure. Then, a robust preconditioner for the chemistry ODE Jacobian matrix is defined, as the result of the optimization of a general-purpose preconditioner-based incomplete LU factorization [37, 38]. Integration of ignition delay calculations at conditions relevant to practical combustion systems is compared for the range of reaction mechanisms chosen, at both solver techniques and with different BDF-based stiff ODE integrators. Finally, the direct and Krylov subspace solvers are compared for modeling practical internal combustion engine simulations.

Mechanism	$\mathbf{n_s}$	$\mathbf{n_r}$	**fuel**	**composition**	**ref.**
ERC n-heptane	29	52	n-heptane	$[nC_7H_{16} : 1.0]$	[39]
ERC PRF	47	142	PRF25	$[nC_7H_{16} : 0.75, iC_8H_{18} : 0.25]$	[40]
Wang PRF	86	392	PRF25	$[nC_7H_{16} : 0.75, iC_8H_{18} : 0.25]$	[41]
ERC multiChem	128	503	PRF25	$[nC_7H_{16} : 0.75, iC_8H_{18} : 0.25]$	[42]
LLNL n-heptane (red.)	160	1540	n-heptane	$[nC_7H_{16} : 1.0]$	[43]
LLNL n-heptane (det.)	654	2827	n-heptane	$[nC_7H_{16} : 1.0]$	[44]
LLNL PRF	1034	4236	PRF25	$[nC_7H_{16} : 0.75, iC_8H_{18} : 0.25]$	[44]
LLNL mehtyl-decanoate	2878	8555	methyl-decanoate	$[md : 1.0]$	[34]
LLNL n-alkanes	7171	31669	diesel surrogate	$[nC_7H_{16} : 0.4, nC_{16}H_{34} : 0.1, nC_{14}H_{30} : 0.5]$	[45]

Table 1: Overview of the reaction mechanisms used for testing in this study.

Description	**ODE integration method**	**order**	**Linear system solution solution**	**Preconditioning**
LSODE, direct	fixed-coefficient BDF	variable	direct, dense	no
LSODES, direct sparse	fixed-coefficient BDF	variable	direct, sparse	no
LSODKR, Krylov	fixed-coefficient BDF	variable	scaled GMRES	ILUT
VODE, direct sparse	variable-coefficient BDF	variable	direct, sparse	no
VODPK, Krylov	variable-coefficient BDF	variable	scaled GMRES	ILUT
DASSL, direct sparse	DAE form, variable-coefficient BDF	variable	direct, sparse	no
DASKR, Krylov	DAE form, variable-coefficient BDF	variable	scaled GMRES	ILUT

Table 2: Overview of the chemical kinetics ODE solution strategies compared in this study.

2. Description of the numerical methodology

2.1. A sparse, analytical Jacobian framework for chemical kinetics

Significant advancements in combustion chemistry codes have occurred in the last few years, but comparison with earlier implementations [46] is not a straightforward task. Often access to the source codes is unavailable, and insufficient description of the numerical algorithms or to the architecture-tailored optimizations is provided. Furthermore, the size of the chemical kinetics problem is typically for small-medium systems ($n_s \leq 10^4$) [47], which makes the adoption of advanced parallel algorithms quite inefficient. Thus, meaningful comparisons ofdifferent numerical methods should be done adopting a same code framework, with the same level of optimization, on the same computing architecture.

The `SpeedCHEM` chemistry code [18] is a computationally efficient chemical kinetics library developed by the authors. The library, written in Modern Fortran standard [48], and compiled with the portable Fortran frontend of the GNU suite of compilers [49], exploits a sparse, analytical formulation for all the chemical and thermodynamic functions involved in the description of the reacting system. The library includes thermal property calculation, mixture states, and reaction kinetics, for the analytical calculation of the Jacobian matrix associated with the chemical kinetic ODE system. Internal BLAS routines have been specifically developed in order not to rely on architecture-dependent or copyright-protected libraries, while exploiting the vectorization features of the language for most low-level code tuning, and leaving most of architecture-related optimization to the compiler.

Still a number of low-level optimizations are needed to enable efficient evaluation of the functions that describe the evolution of the species in the reacting environment, as well as their derivatives. Polymorphic procedure handling has been implemented to enable compliance with a number of different implicit stiff ODE integrators, including VODE [26], LSODE [27], DASSL [28], RADAU5 [5], RODAS [5]. Where not originally available, capability for a direct, sparse solution of the linear system associated with the implicit ODE integration step has been implemented using the code's internal sparse algebra library.

Chemical Kinetics problem. We refer to the *Chemical Kinetics problem* as the initial value problem (IVP) where the laws that define temporal evolution of

mass fractions of a set of species in a reactive environment are described as the result of the reactions involving them. Considering a set of n_s species, $i = 1, ..., n_s$ and n_r reactions, $j = 1, ..., n_r$, every generic reaction is defined as the interaction between a set of reactant species and a set of products:

$$\sum_{i=1}^{n_s} \nu'_{j,i} M_i \rightleftharpoons \sum_{k=1}^{n_s} \nu''_{j,k} M_k, \; j = 1, ..., n_r; \tag{1}$$

where M_i represents species names, and matrices ν' and ν'' contain the stoichiometric coefficients of the species involved in the reactions. As the number of species involved in every reaction is typically small, both these matrices can be extremely sparse. Every reaction features a reaction progress variable rate, q_j, that defines, for the given thermodynamic state of the reactive system, its tendency towards (forward, q_f) producing new products or (backward, q_b) producing reactants:

$$q_j = q_{f,j} - q_{b,j} = k_{f,j} \prod_{i=1}^{n_s} \left(\frac{\rho Y_i}{W_i} \right)^{\nu'_{j,i}} - k_{b,j} \prod_{i=1}^{n_s} \left(\frac{\rho Y_i}{W_i} \right)^{\nu''_{j,i}}, \tag{2}$$

where $k_{f,j}$ and $k_{b,j}$ are reaction rate constants, of the type either $k = f(T)$, $k = f(T, p)$ or $k = f(T, p, \mathbf{Y})$, depending on the complexity of the reaction formulation. The products that involve species concentrations C_i are calculated using mass fractions Y_i, molecular weights W_i, and reactor density ρ. The rate of change of every species' mass fraction is given by:

$$\frac{dY_i}{dt} = \dot{Y}_i = \frac{W_i}{\rho} \sum_{j=1}^{n_r} \left[\left(\nu''_{j,i} - \nu'_{j,i} \right) q_j \right]. \tag{3}$$

Mass conservation, $\sum_i \dot{Y}_i = 0$ or $\sum_i Y_i = 1$, is guaranteed if atom balance in every reaction is verified, i.e., $\sum_i \nu'_{j,i} E_{k,i} = \sum_i \nu''_{j,i} E_{k,i} \forall j = 1, ..., n_r, k = 1, ..., n_e$, where the matrix $E_{k,i}$ contains the number of atoms of element k in species i. While the chemical kinetics equations can be incorporated in any reactive environment formulation, a constant-volume reactor is used, as this is the assumption that is typically adopted when doing operator splitting in multi-dimensional combustion codes [50–53], or for ignition delay calculations in shock tubes:

$$\frac{dT}{dt} = \dot{T} = -\frac{1}{\bar{c}_v} \sum_{i=1}^{n_s} \frac{U_i \dot{Y}_i}{W_i}. \tag{4}$$

Here, $\bar{c}_v$ represents the mass-based mixture averaged constant volume specific heat, and $U_i = U_i(T)$ are the molar internal energies of the species at temperature T. Note however that the approach is also applicable to the sparser, constant-pressure reactor formulation [19].

Sparse matrix algebra. Elementary reactions according to the theory of activated complex represent the probability that collisions among molecules result in changes in the chemical composition of the colliding molecules [54]. Practical combustion mechanisms can incorporate third bodies, but most reactions typically follow the common initiation, propagation, branching, termination pathway structure. This means that, while most species interact with the hydrogen and oxygen radicals of the hydrogen combustion mechanism, large hydrocarbon classes have typically just few interactions among themselves. Looking at the Jacobian matrix $\mathbf{J}$ of different reaction mechanisms, as represented in Figure 1, it is clear that the basic hydrogen combustion mechanism, that typically involves the 10-20 smaller species depending on the formulation [55], is almost dense, and can represent a significant part of the overall reaction mechanism especially when it is very skeletal. However, as more hydrocarbon classes are included in the mechanism, the overall sparsity increases significantly, and can be more than 99.5% for the large reaction mechanisms.

Accordingly, a sparse matrix algebra library was coded to model the stoichiometric index matrices $\nu', \nu''(n_r \times n_s)$, the element matrix $\mathbf{E}(n_s \times n_e)$, the ODE system Jacobian matrix $\mathbf{J}(n_s + 1 \times n_s + 1)$, as well as all the internal calculation matrices involving analytical derivatives such as $\partial\mathbf{q}/\partial\mathbf{Y}$, for instance. The Compressed Sparse Row (CSR) format [56, Chapt. 2] was adopted as the standard to model matrix-vector calculations, matrix-matrix calculations, as well as other overloaded operator functions including sum, scalar-matrix multiplication, comparison and assignment. Even with dynamically allocated matrix structures, the sparsity structures of the matrices are only calculated once at the reaction mechanism initialisation, as they do not change. It was chosen not to consider any dynamic reaction mechanism reduction, to avoid any additional error in the solution procedure. Finally, full encapsulation of the overloaded methods allows their easy implementation in other architectures such as GPUs.

As most ODE integrators for solving stiff problems perform implicit inte-

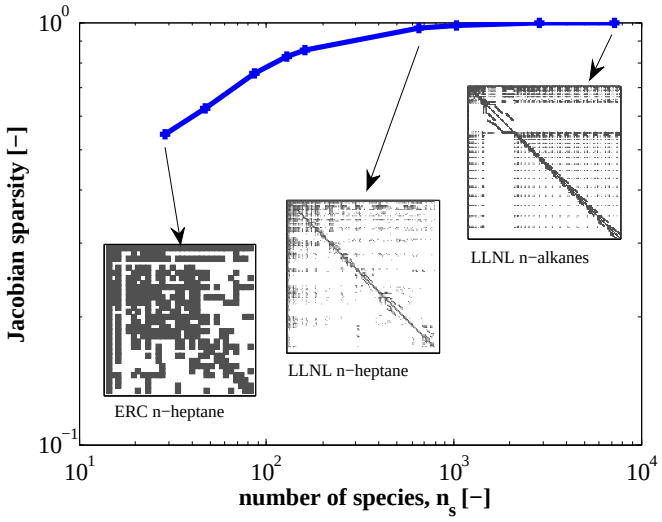

Figure 1: Sparsity of the Jacobian matrix versus reaction mechanism dimension.

gration steps that require evaluation of the Jacobian matrix and the solution of a linear system, or its composition/scaling with some diagonal matrix (see, e.g., [5, §8.8]), routines that perform direct solution of a sparse linear system were implemented using a standard four-step solution method: 1) optimal matrix ordering using the Cuthill-McKee algorithm [57, 58, §4.3.4], 2) symbolic sparse LU factorization, 3) numerical LU factorization, 4) solution by forward and backward substitution. The first two steps of the procedure are only evaluated at the problem initialization, as they only involve the matrix sparsity pattern.

Figure 2 shows the effect of optimal matrix reordering to reduce fill-in during matrix factorization of a large reaction mechanism. Even if the original sparsity of the Jacobian matrix is extremely high, the LU decomposition destroys almost all of it when rows and columns are not correctly ordered. The fill-in phenomenon, i.e. the number of elements in a matrix that switch from having a zero value to a non-zero value after application of an algorithm [59], cannot be avoided when using direct solution methods, and tends to become significant for large reaction mechanisms. However, if the matrix is properly ordered, the number of non-zero elements in the LU decomposition can be not bigger than 2-3 times the number of non-zero elements in the non-decomposed matrix. After the Jacobian matrix has been decomposed into a

lower and an upper triangular matrix, there is no need for matrix inversion as the linear system can be eventually solved in two steps by simple forward and backward substitution:

$$\mathbf{A}x = b \longrightarrow \mathbf{LU}x = b \longrightarrow y = \mathbf{L^{-1}}b \longrightarrow x = \mathbf{U^{-1}}y. \tag{5}$$

Maximizing the use of sparse algebra, and consistently using the same CSR format for all calculations throughout the code, including for building the analytical Jacobian matrix, allows efficient memory usage and minimization of cache misses, that have an impact on the total CPU time. This provides a universal formulation that does not require reaction-mechanism-dependent tuning, and avoids the use of symbolic differentiation tools or reaction-mechanism-dependent routines that would require recompilation of the code for every different reaction mechanism used.

Optimal-degree interpolation. Optimal-degree interpolation was implemented as a procedure to speed-up the evaluation of thermodynamic functions that involve both species and reactions. Most thermodynamic- and chemistry-related are temperature-dependent functions, and include powers, exponentials, fractions, that require many CPU clock cycles to be completed [2]. Functions describing reaction behavior span from simple reaction rate formulations such as the universally adopted extended Arrhenius form, to equilibrium constant formulations that are required to compute backward reaction rates, to more complex reaction parameters such as the fall-off centering factor in Troe pressure-dependent reactions [60]. Functions describing species and mixture thermodynamics, such as the JANAF format laws adopted in the present code, typically feature polynomials of temperature powers and logarithms.
As memory availability is not a restriction in modern computing architectures, caching values of these functions at discretized temperature points, and interpolating using simple interpolation functions, proved to be extremely efficient.

Also the derivatives with respect to temperature were evaluated analytically and tabulated at initialization at equally-spaced temperature intervals of $\Delta T = 10K$. This tabulation allows different degrees of the interpolation, to be calculated only on the basis of the relative position of the current temperature value within the closest neighboring points [18]:

$$f\left(T\right) = \left(T_{n+1} - T\right)/\left(T_{n+1} - T_n\right) = \left(T_{n+1} - T\right)/\Delta T. \tag{6}$$

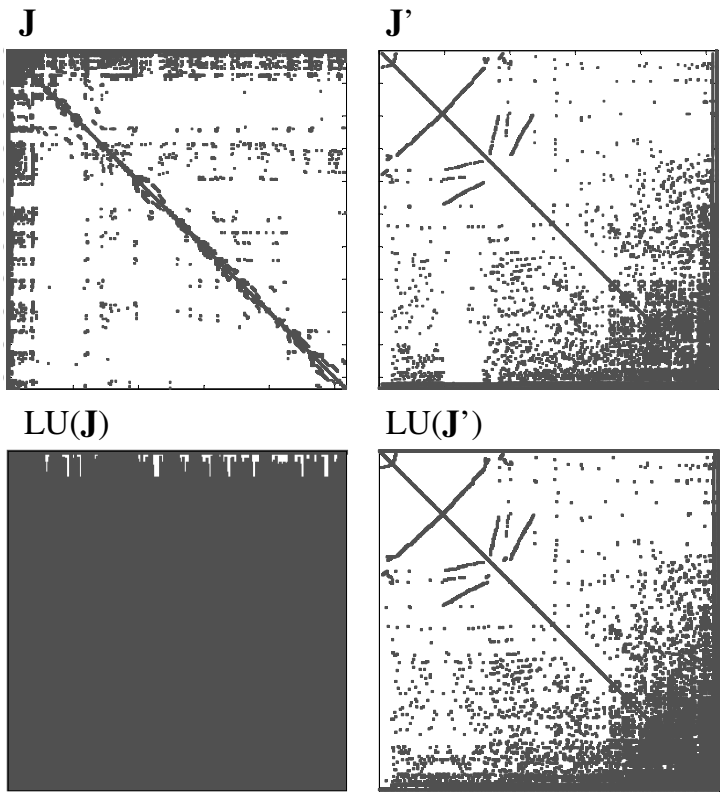

Figure 2: Sparsity patterns of the Jacobian matrix of the LLNL PRF mechanism [44], non-permuted ($\mathbf{J}$) and permuted ($\mathbf{J}'$) using the Cuthill-McKee algorithm, and of their sparse LU decompostions.

At the initialization of every function's table, an optimal degree of interpolation is cached, defined as the minimum interpolation degree that allows the interpolation procedure to predict the function's values within a required relative accuracy, set as $\varepsilon_{tab} \leq 10^{-6}$ in this study, corresponding to 6 exact significant digits. The optimal-degree interpolation procedure also prevents Runge's oscillatory phenomenon by imposing an accuracy check on the interpolated function.

Performance comparisons between interpolated and analytically evaluated thermodynamic functions are shown in Figure 3 using the reaction mechanisms in Table 1. The speed-up factor achieved in CPU time when calling different thermodynamics-related functions using the tabulation-interpolation procedure rather than their analytical formulation is plotted against the number of species in the reaction mechanism, n_s. The plot shows that significant speed-ups of up to about two orders of magnitude are achieved for the most complex reaction properties, even for equilibrium-based reversible reactions and Troe fall-off reactions, whose number in a reaction mechanism can be much smaller than the total number of reactions. The speed-up achieved in evaluating species thermodynamic functions, whose nature is already of a polynomial formulation in the JANAF standard, is smaller but quickly adds up to about 2 times for practical reaction mechanisms.

Overall code performance. Computational scaling of the most expensive functions associated with the integration of the chemical kinetics problem is reported in Figure 4. The results, taken on a desktop PC running on an Intel Core i5 processor with 2.40 GHz clock speed, and with 667 MHz DDR3 RAM, show less than linear computational performance scaling with the problem dimensions of both the ODE vector function r.h.s. and of the Jacobian matrix, and approximately linear scaling, $O(n_s)$, for the complete sparse numerical factorization and solution of the linear system associated with the Jacobian matrix itself. The significant computational efficiency achieved through scalability and low-level optimization of the sparse analytical Jacobian approach is also evident as the CPU time required by Jacobian evaluation is always only between 5.4 and 12.0 times greater than the time needed for evaluating the ODE system r.h.s. throughout all the reaction mechanisms tested, without any changes to the compiled code. With a standard finite difference approach computational cost is estimated to be between 29 and 7200 times greater than the r.h.s. evaluation for the smallest and the largest reaction mechanisms in the group of Table 1. Note that the number of reactions

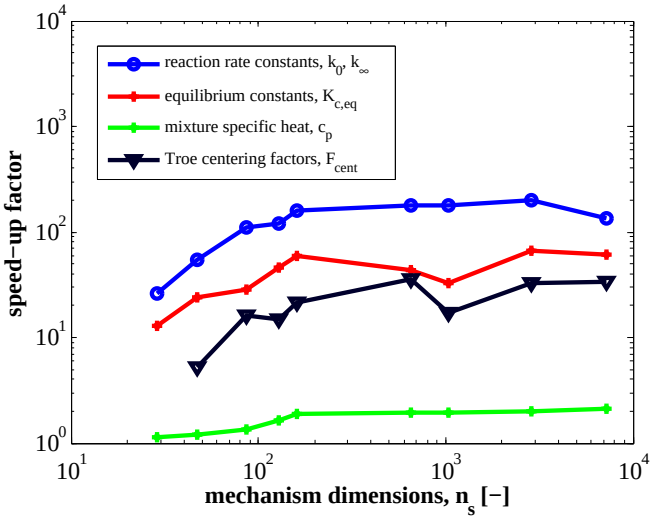

Figure 3: Performance scaling of optimal-degree interpolation vs. analytical evaluation of some temperature-dependent functions.

plays a direct role in the CPU time requirements of the ODE system function and the analytical Jacobian, due to presence of reaction-based functions and sparse matrix-vector (spMxV) multiplications involving $(n_r \times n_s)$ matrices. However, Figure 4 shows that the cost of solving the linear system with the Jacobian, which is species-based, is usually dominant. Furthermore, in order to have a number of reactions large enough to break the linear scaling of the sparse algorithms on the Jacobian, n_r would have to scale at least with $O(n_s^2)$, which would unrealistically correspond to having about $\approx 10^6$ reactions per $\approx 10^3$ species.

The code results were previously validated against both homogeneous reactor simulations and multi-dimensional RANS simulation of engine combustion [18, 35], where it was shown that no differences could be observed between the SpeedCHEM solution and the solution obtained using a reference research chemistry code, when using the same integration method and tolerances.

Profiling of ignition delay calculations showed how much of the total CPU time was spent among the most important tasks, as represented in Figure 5. The linear system solution was still the most computationally demanding task across all mechanism sizes, requiring from roughly 40% up to more than 80% of the total CPU time as the problem dimensions increase. All the other computationally demanding tasks, such as thermodynamic property evalu-

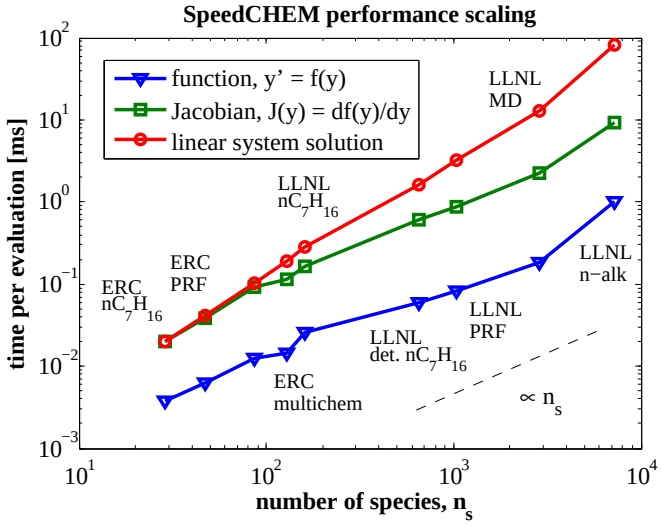

Figure 4: Computational time cost of the functions associated with the ODE system for different reaction mechanisms: r.h.s, Jacobian matrix, direct solution of the linear system (numerical factorization, forward and backward substitution).

ations, sparse matrix algebra, law of mass action, that make up the most expensive parts of the r.h.s and Jacobian function calls, demanded less than 20% of the total CPU time, with the rest of the time devoted to internal ODE solver operation, as well as dynamic memory allocation.

Finally, RAM memory requirements were found to scale slightly less-than-linearly with mechanism dimensions, as an effect of increased Jacobian sparsity for larger hydrocarbon dimensions. Tabulation of temperature-dependent functions appears to be the most storage-demanding feature, whose requirements linearly increase with the number of species (for all the mechanisms tested, the numbers of reactions always followed Lu and Law's rule-of thumb, $n_r \approx 5n_s$, [2]). However, only about $86MB$ of RAM were needed by the SpeedCHEM solver at the largest mechanism. Considering that tabulated data requirements undergo read-only access during the simulations and are shared when parallel multi-dimensional calculations are run, these requirements appear to be negligible compared to current memory availability, even on personal desktop machines.

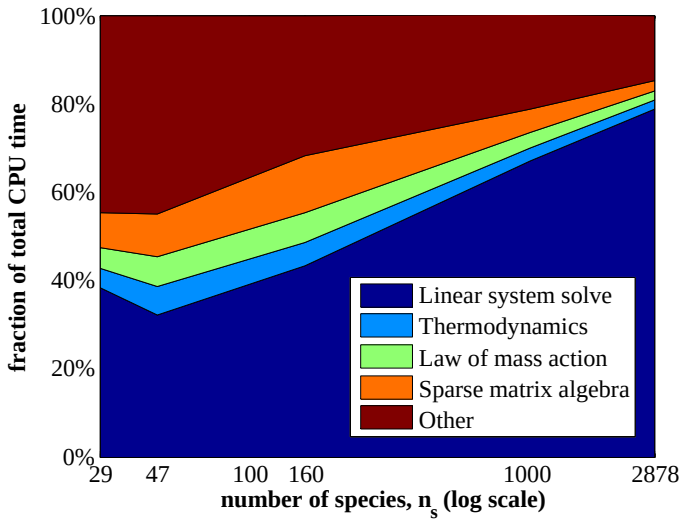

Figure 5: Subdivision of total CPU time among internal procedures during time integration of different reaction mechanisms.

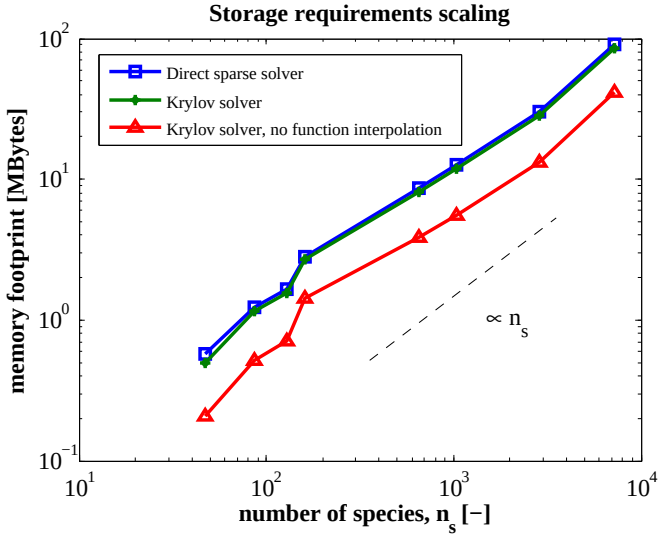

Figure 6: RAM storage requirements of the SpeedCHEM code with and without tabulated data for optimal-degree interpolation. LSODE solver storage featured.

3. Solution of the chemistry ODE Jacobian using Krylov subspace methods

Krylov subspace methods represent a class of iterative methods for solving linear systems that feature repeated matrix-vector multiplications instead of using matrix decomposition approaches of the stationary iterative methods (see, e.g., [47, 61]). These solvers find extensive applications in problems where building the system solution matrix is expensive, or the matrix itself is even not known, or where sufficient conditions [62, Chapt. 7] for the convergence of the iteration matrix of the stationary iterative methods are not known. Their practical applicability is however constrained to special matrix properties, such as symmetry, or to the need to find appropriate preconditioning of the linear system matrix, to guarantee convergence of the method [61, Chapt. 2,3]. Iterative methods such as GMRES [63] are guaranteed convergence to the exact solution in n iterations, n being the maximum Krylov subspace dimension, equal to the linear system dimensions. Thus, if no strategy to accelerate their convergence rate towards the solution within a reasonable tolerance is used, their computational performance can be bad. Furthermore, round-off and truncation errors intrinsic to floating point arithmetic can make the iterations numerically unstable, especially when the problem is ill-conditioned, i.e., its solution is extremely sensitive to perturbations. These problems have limited their implementation in widely available ODE integration packages.

3.1. ODE integration procedure

Backward differentiation formulae (BDF) methods [64] are amongst the most reliable stiff ODE integrators currently available, due to their ability to allow large integration time steps, even in presence of steep transients and the simultaneous co-existence of multiple time scales. The methods exploit the past behavior of the solution for improving the prediction of its future values. BDF methods are incorporated in many software packages for the integration of systems of ordinary differential equations (ODE) and differential-algebraic equations (DAE), including the LSODE package, the base integrator in this study. Given a general initial value problem (IVP) for a vector ODE function $\mathbf{f}$ in the unknown $\mathbf{y}$ of the form:

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}),\ \mathbf{y}(t = 0) = \mathbf{y_0}, \tag{7}$$

BDF methods build up the solution at successive steps in time, $t_n, n = 1, 2, ...,$ based on the solution at q previous steps $t_{n-j}, j = 1, 2, ..., q$, with the following scheme:

$$\mathbf{y_n} = \mathbf{y}(t_n) = \sum_{j=1}^{q} \alpha_j \mathbf{y_{n-j}} + h\beta_0 \dot{\mathbf{y}}_\mathbf{n}, \tag{8}$$

where the number of previous steps, q, also defines the order of accuracy of the method, α_j and β_0 are method-specific coefficients that depend on the order of the integration, and $h = \Delta t = t_n - t_{n-1}$ is the current integration time-step. These methods are implicit as the evaluation of the ODE system function at the current timestep, $\dot{\mathbf{y}}_\mathbf{n} = \mathbf{f}(t_n, \mathbf{y_n})$, is needed to derive the solution. Thus, a nonlinear system of equations has to be solved, which accounts for most of the total CPU time during the integration:

$$\mathbf{F_n}(\mathbf{y_n}) = \mathbf{y_n} - h\beta_0 \mathbf{f}(t_n, \mathbf{y_n}) - \mathbf{a_n} = 0, \tag{9}$$

where $\mathbf{a_n}$ is the sum of terms involving the previous time steps, $\mathbf{a_n} = \sum_{j=1}^{q} \alpha_j \mathbf{y_{n-j}}$. VODE and DASSL incorporate variable α_j and β_0 coefficients whose values are updated based on the previous step sizes to allow for more effective re-usage of the previous values of the solution. However, the fixed coefficient form is more desirable in this study. LSODE handles the solution in Equation (9) through a variable substitution, $\mathbf{x_n} = h\dot{\mathbf{y}}_\mathbf{n} = (\mathbf{y_n} - \mathbf{a_n})/\beta_0$, as follows:

$$\mathbf{F_n}(\mathbf{x_n}) = \mathbf{x_n} - h\mathbf{f}(t_n, \mathbf{a_n} + \beta_0 \mathbf{x_n}) = 0. \tag{10}$$

This nonlinear system has to be solved using some version of the Newton-Raphson iterative procedure:

$$\begin{aligned}
\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{1})} &= \text{initial guess}; && (11)\\
\forall m &= 1, 2, ..., (\text{until convergence}) :\\
\mathbf{F}'_\mathbf{n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}) &= \mathbf{I} - h\beta_0 \mathbf{J}(t_n, \mathbf{a_n} + \beta_0 \tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}), && (12)\\
\text{solve: } \mathbf{F}'_\mathbf{n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})})\mathbf{r_n} &= -\mathbf{F_n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}), && (13)\\
\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m+1})} &= \tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})} + \mathbf{r}_\mathbf{n}^{(\mathbf{m})};
\end{aligned}$$

where the procedure is typically simplified by assuming that the Jacobian matrix does not change across the iterations (simplified-Newton method),

thus $\mathbf{F}'(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}) \approx \mathbf{F}'(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{1})})$, and Equation (13) provides the actual linear system that has to be solved throughout the whole time integration process. The first guess $\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{1})}$ for the solution is estimated applying an explicit integration step (predictor), which is then progressively modified towards the correct solution by the Newton procedure (corrector).

3.2. Iterative linear system solution

The introduction of preconditioned Krylov subspace linear system solution methods into a so-called Newton-Krylov iteration, enters in Equation (13) without modifying the structure of the iterative Newton procedure, as the solution found by the Krylov subspace method is some approximation of the correct solution, and satisfies:

$$\mathbf{F}'_\mathbf{n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})})\mathbf{r_n} = -\mathbf{F_n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}) + \boldsymbol{\varepsilon_n}, \tag{14}$$

where the residual $\boldsymbol{\varepsilon_n}$ represents the amount by which the approximate solution $\mathbf{r_n}$ found by the Krylov method fails to match the Newton iteration equation. It has been demostrated [65] that a simple condition, such as the imposition of a tolerance constraint, $\boldsymbol{\varepsilon_n} < \delta$, is enough to obtain any desired degree of accuracy in $\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}$ at the end of the Newton procedure. The solution $\mathbf{r_n}$ of Equation (14) can be seen as a step of the Inexact Newton method [66] if it satisfies

$$\|\boldsymbol{\varepsilon_n}\| \leq \eta_n \left\|\mathbf{F_n}(\tilde{\mathbf{x}}_n^{(m)})\right\| \tag{15}$$

for a vector norm $\|.\|$. The parameter η_n is called a *forcing term* and it plays a crucial role to avoid the oversolving phenomenon, i.e., the computation of unnecessary iterations of the inner iterative solver (such as the Krylov method) [67]. The convergence of Inexact Newton methods is guaranteed under standard assumptions for $\mathbf{F}$ (see, e.g., [61, pp. 68,90],[68, p. 70]).

The Krylov subspace method to iteratively solve the linear system of Eq. (14) was a scaled version of the GMRES solver by Saad [63] in LSODKR. The algorithm solves general linear systems in the form $A\mathbf{x} = \mathbf{b}$, where A is a large and sparse $n \times n$ matrix, and $\mathbf{x}$ and $\mathbf{b}$ are n-vectors. The idea underlying the Krylov subspace methodology is that of refining the solution array by starting with an initial guess $\mathbf{x_0}$, and updating it at every iteration towards a more accurate solution. The linear system can be written in residual form, i.e., the exact solution is a deviation $\mathbf{x} = \mathbf{x_0} + \mathbf{z}$ from the initial estimate

(typically, $\mathbf{x_0} = 0$), and given the initial residual $\mathbf{r_0} = \mathbf{b} - A\mathbf{x_0}$ we have $A\mathbf{z} = \mathbf{r_0}$. If the Krylov subspace $\kappa_k(A, \mathbf{r_0})$ generated at the k-th iteration by the matrix and the residual r.h.s. vector is the one defined by the images of $\mathbf{r_0}$ under the first k powers of A,

$$\kappa_k(A, \mathbf{r_0}) = \text{span}\left\{\mathbf{r_0}, A\mathbf{r_0}, A^2\mathbf{r_0}, ..., A^{k-1}\mathbf{r_0}\right\}, \tag{16}$$

it can be demonstrated that the solution to the linear system lies in the Krylov subspace of the problem's dimension, $\kappa_n(A, \mathbf{r_0})$. Thus, GMRES looks for the closest vector in this space to the solution, as the vector that satisfies the least squares problem:

$$\min_{\mathbf{z} \in \kappa_k(A, \mathbf{r_0})} \|\mathbf{r_0} - A\mathbf{z}\|. \tag{17}$$

The least squares problem of Equation (17) is solved using Arnoldi's method [69], an algorithm that incrementally builds an orthonormal basis for $\kappa_k(A, \mathbf{r_0})$ at every iteration, i.e., a set of orthogonal vectors starting from a set of linearly independent vectors. The least squares problem can be solved in this equivalent, well-defined orthonormal space rather than on the undefined Krylov subspace. Overall, the number of operations per Krylov subspace iteration is expensive, as they involve a minimization process where the number of unknowns, n, is greater than the current dimensions of the process, k. However, this procedure can be effective if convergence to a desired tolerance can be reached in a number of Krylov iterations that is much smaller than the overall dimensions of the linear system. In practice, the convergence rate can only be accelerated to practical numbers of iterations if some sort of preconditioning is used (see, e.g., [70, 71]).

3.3. Preconditioning of the linear system

The practical applicability of preconditioned Krylov subspace methods typically depends much more on the quality of the preconditioner than on the Krylov subspace solver [63]. Preconditioning in the iterative solution implies computing the linear solution on a different, equivalent formulation, i.e., by left-multiplication of boths sides of the equation by a (left) preconditioner matrix:

$$\begin{aligned} P^{-1}A\mathbf{x} &= P^{-1}\mathbf{b}, \textit{or} \\ \bar{A}\mathbf{x} &= \bar{\mathbf{b}}. \end{aligned} \tag{18}$$

The role of applying a preconditioner matrix is thus that of reducing the span of the eigenvalues of A, and substituting it with a new matrix that is possibly closer to identity, $P^{-1} \approx A^{-1}, \bar{A} \approx I$, for which convergence would be in a single step. However, every step of the Newton procedure of Equation (14) will take the following form:

$$[P_n^{-1}\mathbf{F'_n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})})]\mathbf{r_n} = [P_n^{-1}(-\mathbf{F_n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}))] + \boldsymbol{\varepsilon_n}. \tag{19}$$

Thus, an additional direct linear system solution needs to be solved at every Newton iteration in the following form: $\mathbf{c} = P_n^{-1}(-\mathbf{F_n}(\tilde{\mathbf{x}}_\mathbf{n}^{(\mathbf{m})}))$. From Equation (19) it is evident that it is extremely important that the appropriate preconditioner has a much simpler structure than the original matrix, so that its factorization and solution process for the r.h.s. can be kept as simple as possible. But, it has to approximate the inverse matrix sufficiently well so that the convergence of the Krylov procedure is improved in comparison to the non-preconditioned algorithm.

Ultimately, if the problem's eigenvalues span many orders of magnitude or the preconditioner is too rough, the number of iterations needed for the solution may make the method more expensive than a standard, direct solver, and the solver may fail, even if a solution exists. On the other hand, if the preconditioner provides a too accurate representation of the inverse of the linear system matrix, i.e., $P \approx A, P^{-1}A \approx I$, the dimensions of the Krylov subspace, and similarly the method's convergence rate, will tend to reduce, and the cost of building the preconditioned matrix and solving the r.h.s will typically be unviable.

It has been shown that simple preconditioners such as Jacobi diagonal preconditioning are not accurate enough for chemical kinetics problems [72]. Thus, we have chosen the Incomplete LU preconditioner with dual Truncation strategy (ILUT) by Saad [38, 47]. This preconditioner approximates the original sparse matrix A by an incomplete version of its LU factorization. The factorization is incomplete in the sense that not all of the elements that should be included in the L and U triangular matrices are kept in the incomplete representation. This can significantly reduce the problem of fill-in, both making the factorization much sparser and faster to compute, while still maintaining the LU representation of matrix A within a reasonable degree of approximation. Elements are *'dropped off'* the decomposition in case:

1) their absolute value is smaller than the corresponding diagonal element magnitude, by a certain *drop-off* tolerance δ; 2) only the l_{fil} elements with greatest absolute value are kept in every row and every column of the factorization, thus defining a maximum level of fill-in.
It is interesting to notice that preconditioners for chemical kinetics problems can also be based on chemistry-based model reduction techniques such as the Directed Relation Graph (DRG) [29]. In this respect, usage of a full sparse analytical Jacobian formulation in companion with a purely algebraic preconditioner is promising as:

- the chemistry Jacobian matrix already physically models mutual interactions among species. Its matrix's sparsity structure is exactly the adjacency representation of a graph, where the path lengths among species (i.e., the intensity of their links) are the Jacobian matrix elements, so it already contains the pieces of information that are modeled using techniques such as the DRG method [13];

- in fact, dropping elements from the Jacobian matrix decomposition can be interpreted as the effort of removing the smallest contributions where species-to-species interactions are smaller;

- application of element drop-off makes the matrix factorization faster, and its definition does not rely on chemistry-based considerations, such as reaction rate thresholds, whose validity would not be universal; furthermore, it retains its validity also in case the matrix to be factorized is scaled (for example when diagonal terms are added to the Jacobian matrix, such as in Equation (12)).

Figure 7 compares complete and incomplete LU factorizations of the Jacobian matrix for a reaction mechanism with more than 1000 species [44], at a reactor temperature $T_0 = 1500K$, pressure $p_0 = 50bar$, and with uniform species mass fractions $Y_i = 1/n_s, i = 1, ..., n_s$, so that all the elements in the Jacobian matrix were active. The ILUT decompositions have been obtained by keeping a constant maximum level of fill-in, $l_{fil} = n_s$, and using different drop-off tolerance values, $\delta = \{0.1, 10^{-3}, 10^{-6}, 10^{-10}\}$. As the Figure shows, the sparse complete LU decomposition leads to about 95% fill-in rate, featuring 40539 non-zero elements instead of the Jacobian's 20765 non-zeroes. Note that the last line and the last column of the Jacobian matrix, as well

as that of the LU decomposition, are dense, as the energy closure equation does uniformly depend on all of the species.

The ILUT factorization shows instead a significant reduction in the total number of non-zero elements, and at the drop-off value recommended by Saad [38], $\delta = 0.001$, only 22.7% of the elements of the complete LU factorization survive the truncation strategy. The final number of elements monotonically increases when reducing the drop-off tolerance value; even at the smallest value of the threshold, $\delta = 10^{-10}$, where only the values that are at least ten orders of magnitude smaller than their diagonal element are cancelled. Thus, a significant 28.4% of elements are dropped from the complete LU, conveying the extreme stiffness of the chemical kinetics Jacobian.

4. Results

The robustness and the computational accuracy of Krylov subspace methods incorporated in stiff ODE solution procedures are compared with direct solution methods first for constant-volume, homogeneous reactor ignition calculations. The LSODES solver [27] is used for the direct, complete sparse solution of the linear system. An interface subroutine has been implemented to provide the Jacobian matrix, analytically calculated in sparse form using the chemistry library, columnwise as requested by the solver. Even if dense columns are requested by the solver interface, including zeroes, we provide the solver with the sparsity structure of the optimally permuted Jacobian matrix at every call. The direct linear system solution is then computed by LSODES at every step by applying complete sparse numerical LU factorization, and then forward and backward substitution. Symbolic LU factorization, i.e., calculation of the sparsity structure of the L and U matrices, is computed by the solver once per call. The sparsity structure of the analytical Jacobian matrix provided by the `SpeedCHEM` library never changes throughout the integration, as zero values that may arise in case some species are not currently present, are replaced by the smallest positive number allowed by floating-point accuracy (2.2×10^{-308}, in double precision).

The LSODKR version of the LSODE package is used to deal with a preconditioned Newton-Krylov implicit iteration. The LSODKR solver uses SPIGMR, a modified version of the preconditioned GMRES algorithm [63], that also scales the different variables so that the accuracy for every component of the solution array responds to the tolerance formulation of LSODE's integration algorithm:

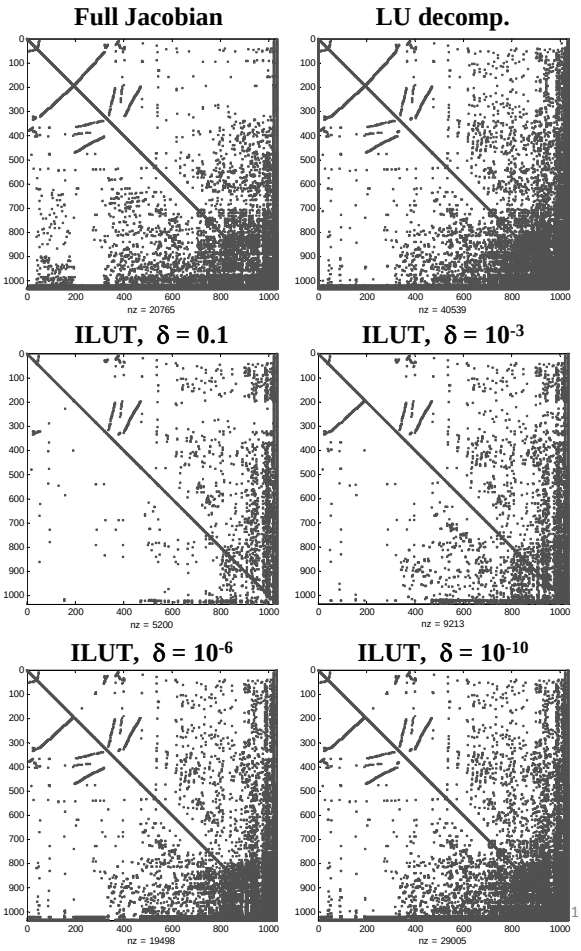

Figure 7: Comparison between sparsity pattern of complete and various incomplete LU decompositions for the Jacobian matrix of a reaction mechanism with $n_s = 1034$ [44].

$$|\tilde{y}_i - y_i| \leq RTOL_i \, |y_i| + ATOL_i. \tag{20}$$

Here, $RTOL_i$ and $ATOL_i$ are user-defineable relative and absolute values, that have been kept constant in this study at 10^{-8} and 10^{-20}, respectively. The LSODKR solver interfaces to the chemistry library in two steps. First, the Jacobian calculation and preprocessing phase is called; Saad's ILUT preconditioner has been coupled to the `SpeedCHEM` library so that preconditioning could be performed at the desired drop-off and fill-in parameter values, and eventually a sparse matrix object containing the L and U parts could be generated. As a second step, a subroutine for directly solving linear systems involving the stored copy of the preconditioner as the linear system matrix has been developed, using the chemistry code's internal sparse algebra library [18]. Thus, LSODKR computes the Jacobian matrix only when the available copy is too old and unaccurate, and then performes multiple system solutions as requested by the r.h.s. of the preconditioned Krylov solution procedure of Eq. (19).

As the critical factor that determines the total CPU time when integrating chemical kinetics is the reaction mechanism size [2], we have investigated a range of different reaction mechanisms, from $n_s = 29$ (skeletal n-heptane ignition for HCCI simulations [39]), to $n_s = 7171$ (detailed mechanism for 2-methyl alkanes up to C20 and n-alkanes up to C16 [45]), as summarized in Table 1. Skeletal or semi-skeletal mechanisms with 20-200 species of interest in multi-dimensional combustion simulations were included. A representative multi-component fuel composition was used to activate more reaction pathways and keep most elements in the Jacobian matrix active. A standardized matrix of 18 ignition delay simulations was established. The initial conditions mimic validation operating points for reaction mechanisms tailored for internal combustion engine simulations and span different pressures $p_0 = [20.0, 50.0]bar$, temperatures $T_0 = [650, 800, 1000]K$ and fuel-air equivalence ratios, $\phi_0 = [0.5, 1.0, 2.0]$. The temporal evolution of all of the 18 initial value problems has been integrated through a same total time interval of $\Delta t = 0.1s$.

4.1. Study of optimal preconditioner settings

A parameter study involving the ILUT preconditioner drop-off tolerance δ and maximum fill-in parameter l_{fil} was conducted to determine the optimal

tradeoff between computational complexity of the incomplete factorization, and the convergence rate of the Krylov solver. The matrix of ignition delay calculations was run serially on a single CPU for every reaction mechanism, spanning drop-off tolerances $\delta = [10^{-3}, 10^{-6}, 10^{-8}, 10^{-10}]$ and maximum fill-in parameters, expressed in terms of fraction of the Jacobian matrix size, $f_{fil} = [0.25, 0.50, 0.75, 1.0]$, where $f_{fil} = l_{fil}/(n_s + 1)$, corresponding to a total of 288 ignition delay simulations per solver.

In Figure 8, the computational performance of the simulations as a function of the two ILUT preconditioner settings is reported. CPU time values are normalized versus the corresponding simulation times requested by running the same ignition delay calculations using LSODES with the direct solver. Finally, response surfaces were reconstructed using ordinary kriging.

The shape of the response surfaces was consistent for most reaction mechanisms. In particular, dependency on the drop-off tolerance order appeared to be much stronger than the maximum fill-in parameter. Only for the smallest and more dense reaction mechanisms, some slight effect of the maximum fill-in parameter was seen when the value was as small as about 25% of the row length, and the total computational time was increased due to the slower convergence rate of the Krylov iterative solver. As for the actual speed-up achievable using the Krylov subspace solver, it appears that no improvement could be obtained for reaction mechanisms smaller than $n_s \approx 650$. The total CPU time required by LSODKR was consistently faster than the direct solver for the largest and sparsest mechanisms. A significant speed-up of about two times was observed only for the largest reaction mechanism, with $n_s = 7171$. More important, coupling of the ILUT preconditioner with the Krylov subspace solver proved to be an extremely robust and efficient methodology, as no solver convergence failures were observed throughout the whole set of simulations, even with tight integration tolerances. Ultimately, optimal settings for the ILUT preconditioner of $\delta = 10^{-8}$ and $f_{fil} = 0.8$ were selected based on the following considerations:

- local minima were consistently observed at drop-off tolerance values not larger than $\delta = 10^{-6}$. Coarser tolerances, such as of the order of $\delta = 10^{-3}$, as suggested by Saad [38], have too slow convergence rates for chemical kinetics problems, and the total simulation time was as much as 3-4 times bigger than with direct system solution;

- minima appeared at different fill-in parameters f_{fil} across different reaction mechanisms. The effect of this parameter on the overall-speed

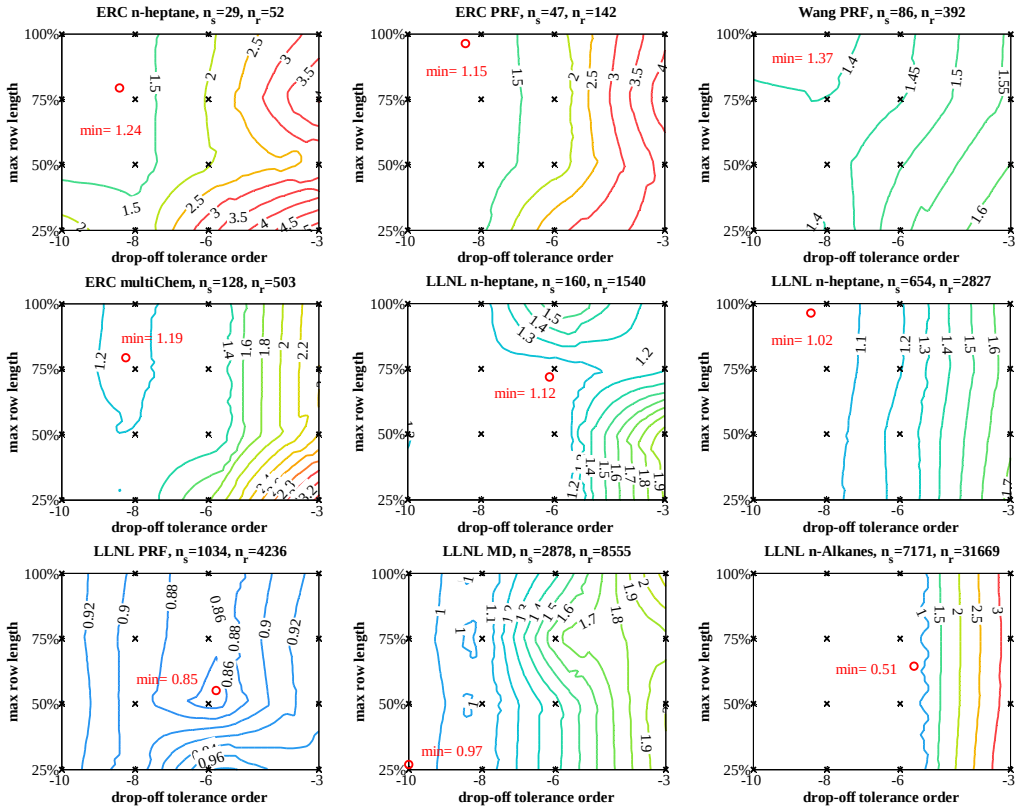

Figure 8: Influence of ILUT preconditioner settings on relative CPU time performance of the preconditioned Krylov solver-enabled integration for different reaction mechanisms. Contours represent normalized CPU times against CPU time using LSODES and direct sparse solver. (x marks) actual measurement points, (lines) response surface contours using kriging, (O marks) interpolated local minima.

up was much less relevant, but in some cases allowing complete rows, i.e., $f_{fil} = 1.0$, was too expensive;

- extreme drop-off tolerances of $\delta = 10^{-10}$ never yielded faster solutions, meaning that a more accurate and expensive LU decomposition was not compensated enough by the faster convergence rate of the Krylov solver.

In Figure 9, cumulative integration metrics are reported for medium and large size mechanisms. Total number of time steps taken, ODE system function and Jacobian matrix evaluations, and linear system solutions are plotted as

a function of the drop-off tolerance value, δ, at the optimal fill-in parameter value $f_{fil} = 0.8$, in comparison with the direct solver metrics. The plots show consistent behavior between the two mechanisms, despite the large difference in size and sparsity. Relevantly, the coarsest drop-off tolerance, $\delta = 10^{-3}$, was extremely unefficient in both CPU time performance and integration metrics, as the roughness of the ILUT decomposition requires the ODE integrator to take an extremely high number of integration steps to achieve the desired solution accuracy. Monotonic reduction of all integration metric parameters was seen as the accuracy of the LU decomposition improved. Interestingly, the LSODKR solver took a smaller number of timesteps to completion than LSODES. However, despite similar numbers of Jacobian matrix evaluations (both solvers share the same Jacobian reusage strategy), the overall number of function evaluations and linear system solutions was significantly higher with the Newton-Krylov iteration. In particular, the average number of Krylov subspace iterations per timestep (estimated equal to the number of calls to the linear system solution subroutine), was between 4.47 and 4.98 for the preconditioner tolerances smaller than 10^{-4}.

The reaction mechanism size did impact the main solution metrics, but it did affect the total CPU time. The small mechanism showed an optimal trade-off point at $\delta = 10^{-8}$, while the large mechanism reached a plateau in CPU time at $\delta \leq 10^{-8}$. The knee when using the large reaction mechanism was smaller, but the CPU time using the optimal tolerance was still smaller than when using the smallest δ, by about 4.1%.

4.2. Overall computational efficiency

4.2.1. Ignition delay calculations

The computational efficiency of the preconditioned BDF-Krylov solver LSODKR was compared to both LSODE, that uses direct methods and dense matrix algebra, with sparsely-evaluated analytical Jacobian converted into dense form, and LSODES, with the previously described configuration. Figure 10 reports the average CPU time averaged over the 18 ignition delay cases versus reaction mechanism size. Interestingly, despite the fact that Krylov solver techniques have been historically developed for large-scale applications, the performance of the Krylov solver was consistently more efficient than the dense solver also for small mechanism sizes, even if the Jacobian matrix was computed in sparse analytical form in that case too. The direct solver outperformed the Krylov solver for all mechanism sizes smaller than about one thousand with greater efficiency at the smallest mechanisms,

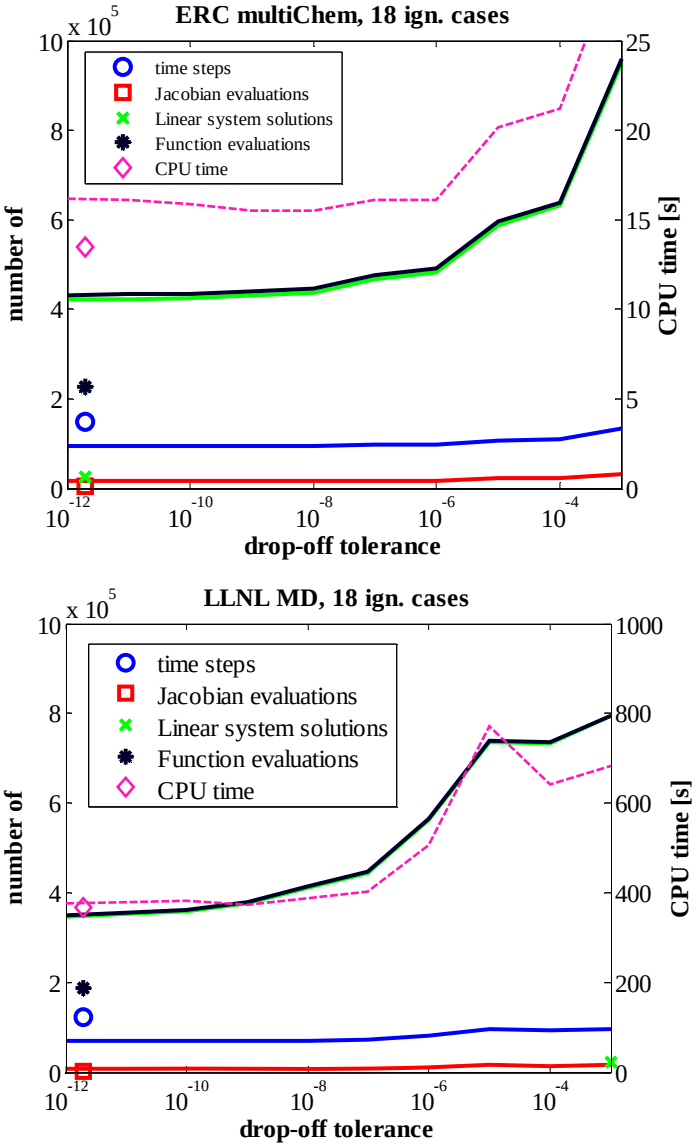

Figure 9: Cumulative integration metrics for 18 ignition delay simulations using the LSODKR solver with ILUT preconditioning, versus ILUT drop-off tolerance δ. (top) ERC multiChem mechanism [42], (bottom) LLNL detailed methyl-decanoate mechanism [34]. Marks represent corresponding metrics for the LSODES solver.

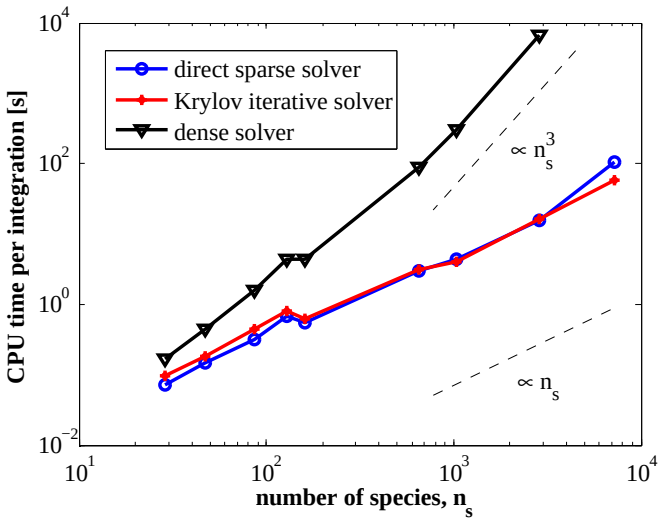

Figure 10: Average CPU times per integration over 18 ignition delay simulation cases, using different linear system solution strategies and the fixed-coefficient LSODE BDF method.

where the Krylov solver required up to about 34% more CPU time. The benefit of the preconditioned Krylov solver was observed only for the largest reaction mechanism size, with $n_s = 7171$, where it outperformed the direct sparse solver by 45.4%.

Integration metrics, plotted against reaction mechanism size in Figure 11, show that the structure, including the sparsity, does not affect the solver behavior in terms of the integration parameters; similarly, the differences in behavior observed between the Krylov and the direct sparse solver are seen throughout all the mechanism sizes. One only significant difference is seen when looking at the number of Jacobian evaluations taken by the direct solver for the largest mechanism, when its exact formulation is calculated to bring the Newton procedure in the implicit step down to convergence.

A closer look at the BDF method is reported in Figure 12, where ignition of a stoichiometric PRF25 fuel-air mixture was simulated using the ERC multiChem mechanism [42], at initial conditions $p_0 = 50bar, T_0 = 800K$, using the LSODE solver. This mechanism was the stiffest due to the extremely lumped reaction pathways for hydrocarbon classes up to C_{16} (cf. Figure 11). Here, the accuracy of all the solutions was not affected by how differently

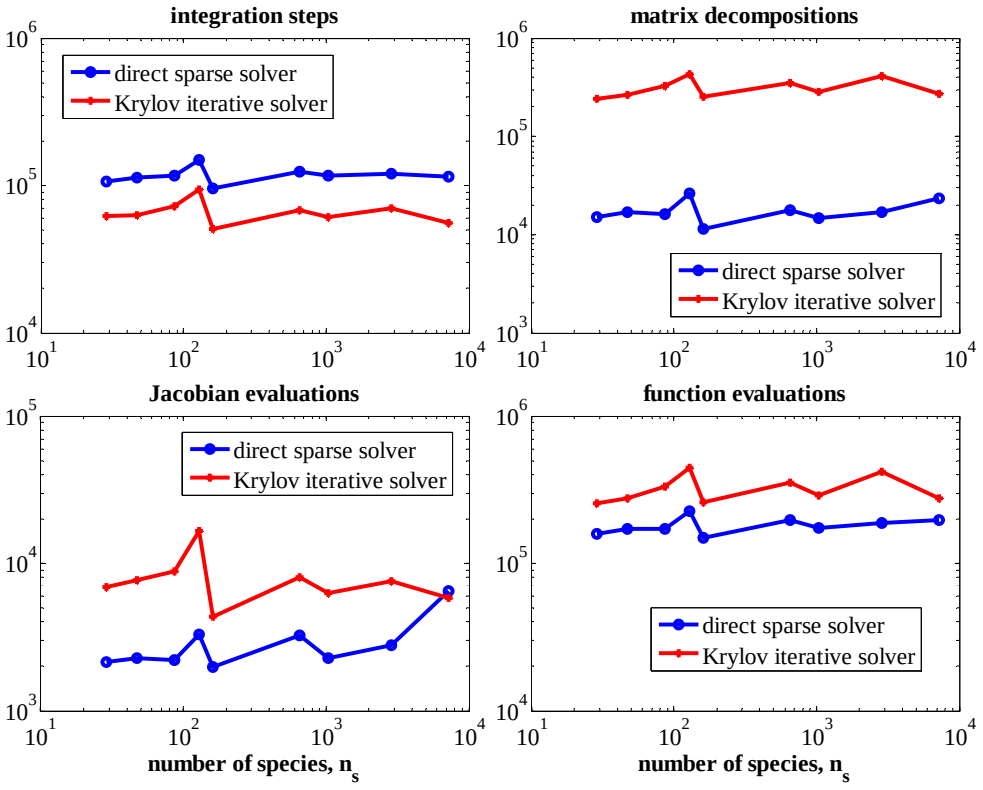

Figure 11: Integration metrics for 18 ignition cases versus reaction mechanism size. (blue) LSODES with direct solver, (red) LSODKR Krylov solver with ILUT($\delta = 10^{-8}, f_{fil} = 0.8$) preconditioning.

the linear system solution was addressed internally within the implicit step iterations. However, slight differences are seen when looking at the time stepping, i.e., at how large time steps the ODE solvers took to compute the solution, as an effect of solution algorithm and round-off and truncation errors arising from the different procedures. 1) During low-temperature ignition ($0.3ms < t < 0.6ms$), the solvers (dense direct, sparse direct and Krylov iterative) show surprisingly similar timestepping, including spikes towards small time step values occurring when the solvers failed to converge due to severe stiffness. 2) during the no-heat-release, preparatory phase when chain initiations lead to generation of extremely small species mass fractions, the Krylov solver kept integrating with larger timesteps while both direct solvers had sudden time step reductions. This suggests that suppression of the smallest elements in the LU decomposition aids the solution convergence when many species have very small mass fraction values. 3) after ignition, when the reactions are near equilibrium and the system's stiffness is at its maximum, again the Krylov solver outperformed both other solvers in terms of both taking larger time steps and of achieving smoother integration (fewer restarts).
Finally, integration performance was studied in comparison with other stiff ODE solvers that adopt preconditioned Krylov subspace methods: VODPK [73] and DASKR [74], the Krylov enabled versions of VODE and DASSL, respectively. Simulations with the VODE and DASSL ODE solvers using direct linear system solution were run too, where the internal Jacobian communication and linear system solution algebra parts were replaced by using `SpeedCHEM`'s sparse matrix library. Figure 13 summarizes the computational performance when the optimal ILUT preconditioner parameters were used. All three solvers (LSODKR, VODPK, DASKR) show consistent, efficient behavior and overall scaling of the order of $O(n_s)$ was achieved, both when using the direct sparse linear system solver, or the preconditioned Krylov method. LSODKR and VODPK in particular showed very similar trends, though VODPK was slightly more efficient for large reaction mechanisms. The implicit formulation in DASKR, that makes it suitable for differential-algebraic (DAE) systems, appeared less efficient when using internal Newton-Krylov iteration, but the version that used `SpeedCHEM`'s sparse matrix algebra implementation, DASSL, was the fastest solver in the group. The variable coefficient formulation in VODE made its behavior different than LSODE when switching from direct sparse algebra to the preconditioned Krylov method: the latter being consistently faster than the direct approach throughout most of the mechanisms tested. LSODE with sparse direct algebra was signif-

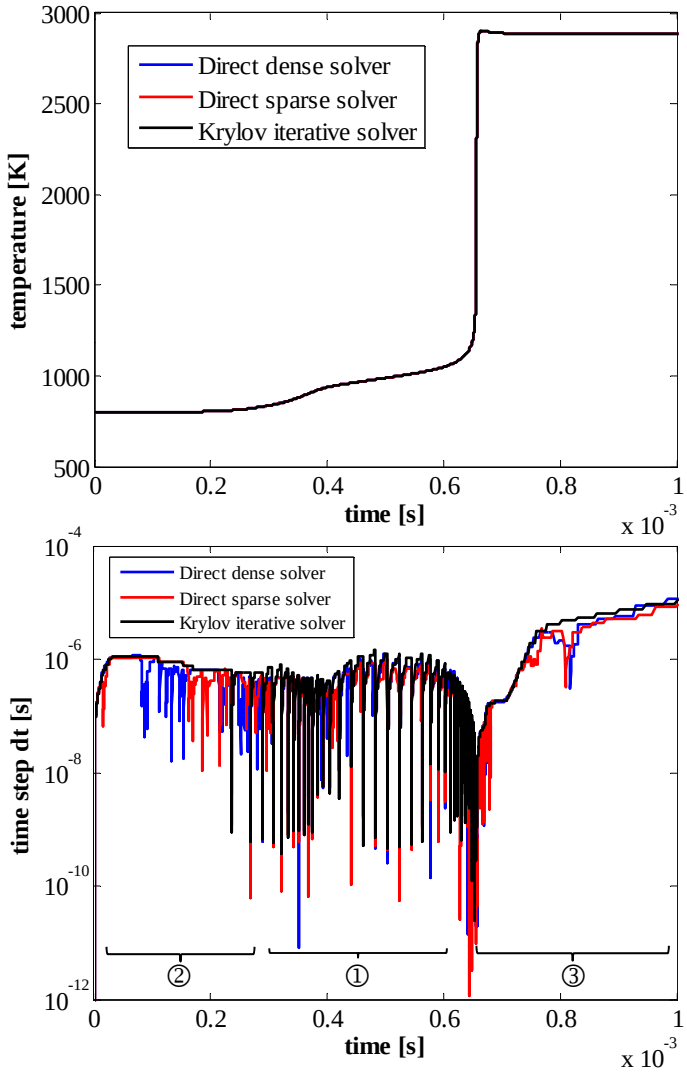

Figure 12: Linear system solver comparison for ignition of a PRF25 fuel using the ERC multiChem mechanism and LSODE method, $\phi = 1.0, p_0 = 50bar, T_0 = 800K$: (top) predicted temperature profile, (bottom) integration time steps.

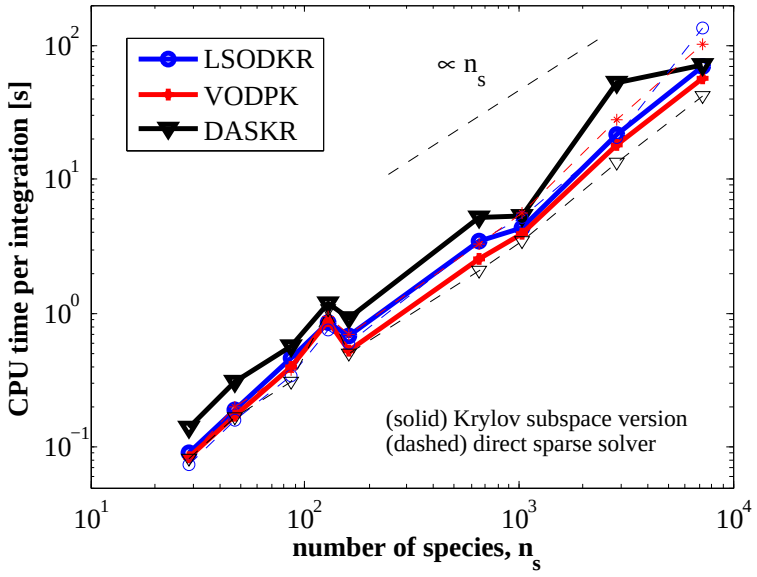

Figure 13: Average CPU times per integration over 18 ignition delay simulation cases, using (solid lines) ILUT preconditioned Krylov subspace solvers LSODKR, VODPK, DASKR; or the corresponding direct solver versions (dashed lines) with custom sparse matrix algebra: LSODES, VODE, DASSL.

icantly faster than its Krylov version for all the reaction mechanism sizes $n_s \leq 654$.

4.2.2. Internal Combustion Engine simulations

As pointed out in a previous study [18], the computational efficiency of stiff ODE solvers is different when incorporating them within operator-splitting calculation frameworks, such as for the integration of chemical kinetics in multidimensional combustion CFD simulations. The KIVA codes [50, 52, 53] model internal combustion engine flows with sprays and chemical reactions. The code makes use of operator-splitting for spray and chemistry terms. The chemistry evolves in every cell of the computational grid independently as in separate, constant-volume adiabatic reactors. The chemistry integration time requested for every cell is thus the fluid flow solver time-step, and due to accuracy constraints of the Arbitrary Lagrangian-Eulerian methodology, this can be as small as $\Delta t = 10^{-7}$ to $10^{-6} s$. This made coupling with the KIVA CFD code a challenging benchmark for the present ODE solu-

tion methods. In fact, typical integration time steps of stiff chemistry ODE solvers accidentally fall in this range, as shown in Figure 12. Thus, their overall computational performance when coupled to a CFD code relies not only in their capability to perform large time steps, but more importantly, in being able to perform very few Newton iterations within the same step, and achieving high order accuracy, as Jacobian matrix and solution reusage for large time-steps are scarcely possible. Techniques are available to reduce the computational burden by reducing the CFD domain size to a limited number of CFD cell 'clusters' or 'zones' [75] from which the solution can be mapped back to the individual cells based on a proximity criterion [17], but at the cost of introducing some error. However, as the aim of this study was to achieve a solution without approximations, high-dimensional cell clustering features were not used.

In order to compare the computational performance of direct and Krylov subspace solvers for internal combustion engine simulations, closed-valve simulations of a light duty optical research engine [76] were chosen. The simulations model a partially premixed combustion strategy, operated through a single, early injection of PRF25 fuel, at a boosted, $55\%EGR$, $3barIMEP$ low-load operating condition. For computational efficiency, a $1/7th$ sector mesh with 18399 cells at bottom dead center was used. A customized version of the KIVA-3v code that incorporates the ERC models for spray injection, atomization, multi-component evaporation, droplet collision and gas-phase interaction, as well as improved turbulence modelling was used. A full description of the code's submodels and of the numerical setup is reported in [77]. Five different operating conditions were studied, including different swirl ratios $R_s = [1.5, 2.2, 3.5]$, and different fuel injection pressure values, namely $p_{inj} = [500, 860, 1220]$ bar. Chemistry was solved using the `SpeedCHEM` library, and the relatively small, skeletal ERC PRF mechanism, which has $n_s = 47$ and $n_r = 142$ [40]. The maximum time-step allowed by the fluid flow solver was $3 \times 10^{-6}s$. All the simulations were run in serial mode on one processor, on a computing cluster featuring AMD Opteron processors with 4 cores per CPU and a clock speed of 2700 MHz.

The CPU time comparison of the simulations for the three solvers LSODE, VODE, DASSL in both Direct and preconditioned Krylov configurations is reported in Figure 14. Specific chemistry times are separated from the rest of the code operation, that includes the fluid flow and spray solution. Clearly,

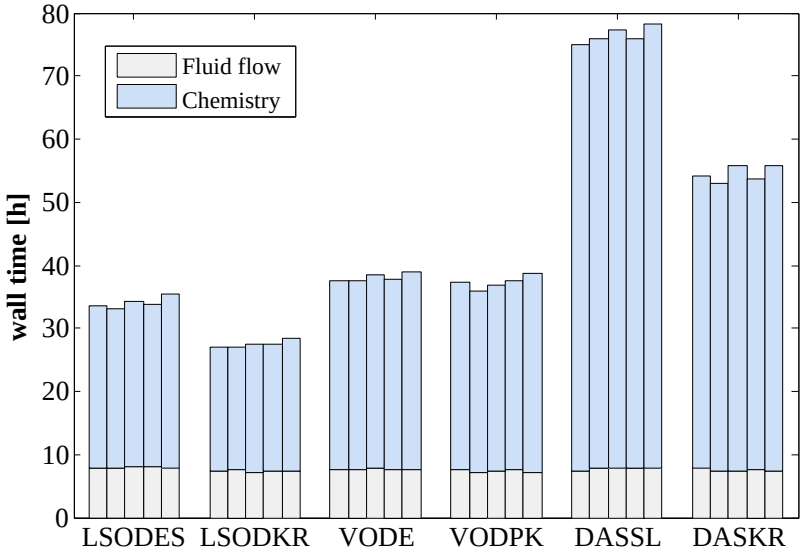

Figure 14: Solver wall-time comparison for 5 KIVA engine case simulations.

both the VODE and the LSODE solvers showed similar overall performance when using the direct solver. Usage of the Krylov method in LSODE provided a significant speed-up of the solution (an average of -23%) . For VODE, the advantage of the direct solver seen for ignition simulations was not seen, but similar performance was observed between the two methods. The DASSL solver showed significant benefits in adopting the Krylov iteration method as LSODE, but its overall computational times were larger than for the other solvers. Noticeably, all solvers performed consistently across all cases, and no Krylov method failures were observed with the optimal preconditioner settings despite the wide variety of reactor conditions. Also, no noticeable differences could be observed in the predicted engine pressure and temperature traces, as shown for example in Figure 15. The overall agreements of all the simulations were within a relative error of 0.63% for the pressure traces. In order to quantify the computational efficiency of the solvers during the simulation, a *grid-index* value was used. This index is defined as the average CPU time spent on every cell of the grid at every time step of the KIVA solver. Taking the simulation with $R_s = 1.5$ and $p_{inj} = 860bar$ as representative, details of the grid index behavior during the simulations are reported

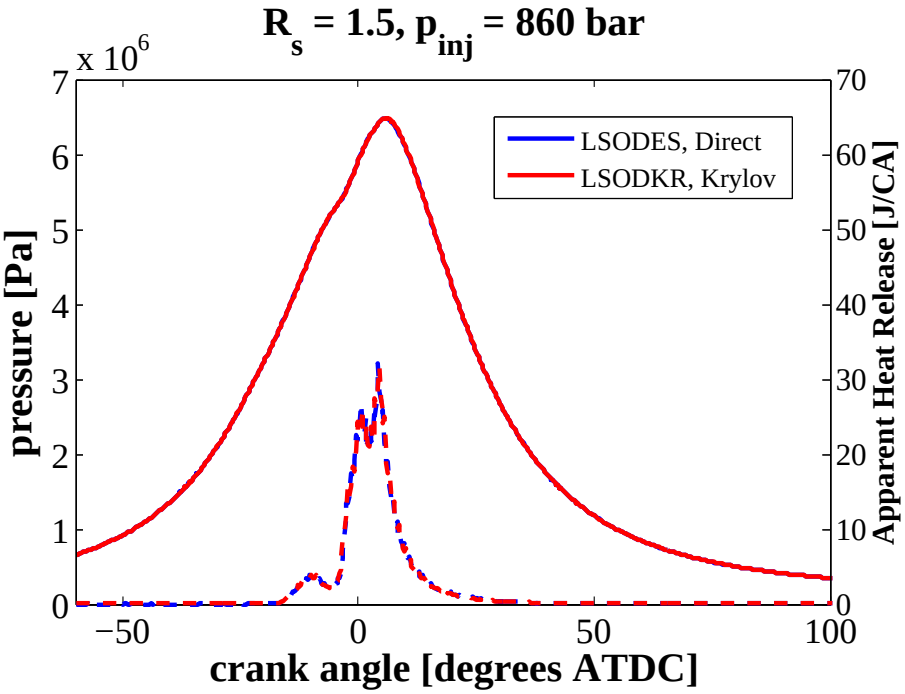

Figure 15: Solution comparison for the $R_s = 1.5, p_{inj} = 860bar$ case. (solid) in-cylinder pressure trace, (dashed) apparent rate of heat release. Chemistry integration using direct vs. preconditioned Krylov LSODE solver.

in Figure 16. Confirming the results of the ignition delay integrations, it is seen that the Krylov method is much more suitable for the integration of the low-temperature region than the direct solvers. Grid index values achieved by the Krylov solvers at crank angles smaller than about $-30°ATDC$ outperformed the corresponding direct solvers by up to more than three times. On the contrary, during high temperature ignition, the Krylov solvers were always outperformed by the direct solvers, except for LSODE, where both approaches had similar computational time requirements. Finally, later after ignition, when the mixture reaches near-equilibrium conditions, and strong, but almost equal, forward and backward reaction contributions compete, the Krylov subspace methods performed similarly, or faster, than the corresponding direct ones.

In conclusion, the present engine simulation study confirmed the results observed for the ignition delay calculations. However, the integrators were forced to small time steps by the CFD solver and required repeated Jacobian evaluations/solutions due to the operator-splitting procedure. Tis enhances differences in performance among the solvers. In particular, the Krylov method significantly outperformed the direct solver in the region of the compression stroke where no fuel is present within the mixture, and the temperature is low, and again late after ignition, where the kinetics are near equilibrium. The direct solvers were instead consistently faster than their Krylov-enabled counterparts throughout the whole fuel injection and combustion duration. Amongst the solvers, both versions of LSODE showed the best performance.

5. Concluding remarks

The computational performance of direct and preconditioned iterative Krylov subspace solvers for the solution of the stiff ODE systems associated with combustion chemical kinetics has been studied. The chemical kinetics of reactive gaseous mixtures was been modeled using the `SpeedCHEM` package, an efficient Fortran library for combustion chemistry that features sparse analytical formulations for the most important functions, optimal-degree interpolation of expensive thermodynamic properties, as well as a full library for internal sparse algebra that handles both matrix and vector computations and interfaces to stiff ODE solvers. The LSODES and LSODKR packages were compared, as they implement the same implicit BDF method

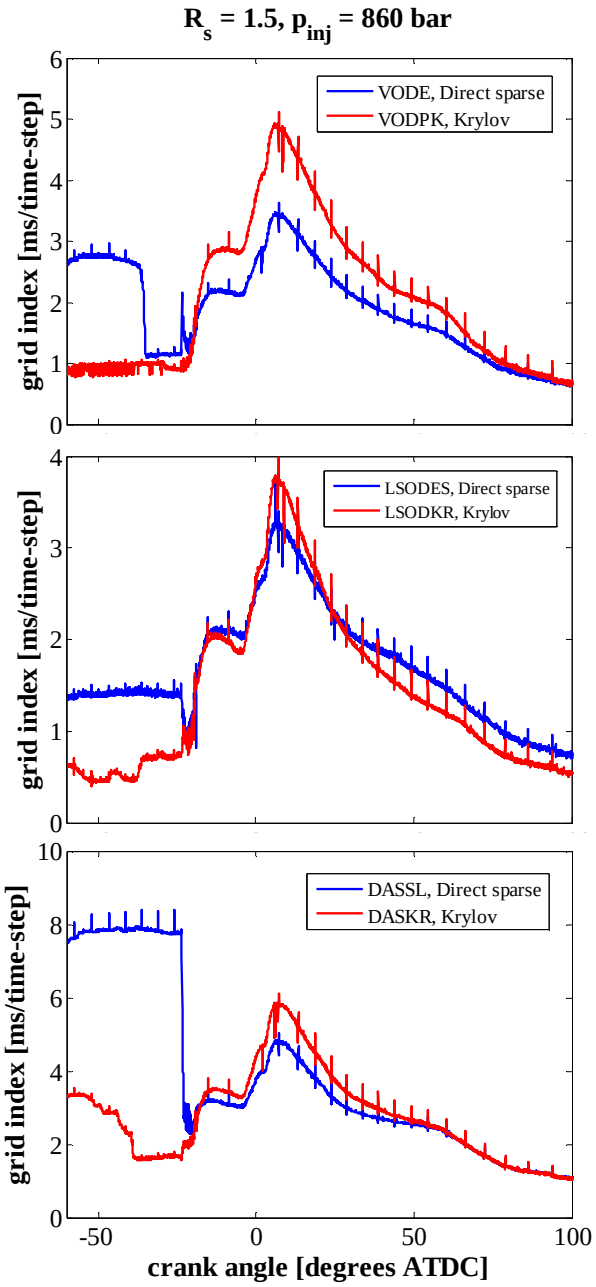

Figure 16: Grid index comparison for the $R_s = 1.5, p_{inj} = 860bar$ case, Direct vs. preconditioned Krylov ODE solver operation: (top) VODE, (center) LSODE, (bottom) DASSL.

and the same Jacobian re-usage strategy. Saad's sparse incomplete *LU* factorization with dual truncation strategy (ILUT) was used as the problem preconditioner, in combination with the SPIGMR solver when the Krylov subspace solution technique was used. Other BDF solvers, such as DASSL and VODE, including their Krylov-enabled versions, DASKR and VODPK, were also compared to the LSODE solver. Comparisons were also made with the `SpeedCHEM` code coupled to the KIVA-ERC CFD code to perform internal combustion engine simulations of a light duty diesel engine operating with a partially premixed combustion strategy.

The study showed that Krylov subspace methods, with a robust, general-purpose preconditioner, are extremely efficient for simulating combustion chemistry problems. The direct solver approach however provided greater computational efficiency for smaller reaction mechanisms that are of practical interest for multidimensional combustion simulations, where $n_s \leq 10^3$. In particular:

- the sparse analytical Jacobian formulation made it possible to find a suitable preconditioner for the integration. The ILUT's strategy that drops the least important elements in the decomposition is similar to removing interactions among species whose mutual links are weak. In this sense, the approach has the same effects as on-the-fly mechanism reduction approaches such as the DRG. The simplification is achieved without using expensive species relation techniques that would make the construction of the reduced Jacobian matrix much more demanding, since the graph links between the species, as well as its structure, would have to be re-calculated at every timestep;

- optimal preconditioner settings of $\delta = 10^{-8}$ and $f_{fil} = 0.8$ were found after an optimization study. These settings suggest that accurate factorization is needed. Despite this accuracy constraint, the final number of non-zeros can be as low as about half that of the complete factorization;

- no solver failures due to bad convergence of the Krylov subspace method were observed when using the ILUT preconditioner;

- integration metrics at the optimal preconditioner settings were close to the metrics of the direct solver method, confiming that the ILUT

is a robust, general-purpose preconditioner, and the optimal settings chosen were appropriate for this class of problems. A smaller number of integration time steps was needed for the Krylov solver;

- the overall performance of the direct solver was consistently faster than the Krylov subspace solver for reaction mechanisms smaller than about 1000 species;

- the dimensions of the chemistry ODE system were evidently not large enough to fully exploit the potential of Krylov solvers;

- when incorporated into multidimensional engine CFD simulations with a skeletal reaction mechanism, the Krylov method was faster than the direct method during the low-temperature, pre-ignition phase, and had similar performance after ignition, at near-equilibrium conditions. During ignition, the direct solvers always outperformed their Krylov-enabled versions;

- a computational speed-up of nearly 25% on chemistry could be achieved by adopting the LSODKR solver in comparison with LSODES, that has already been shown to be the fastest among the direct solvers within CFD simulations. Greater speed-ups could be achieved by adopting a hybrid solution strategy where the most appropriate solver method is chosen based on the local mixture conditions.

Finally, an important finding is that both methods achieve scaling of the computational cost of the solution from $O(n_s^3)$ down to about $O(n_s)$ for the very sparse combustion chemistry models that describe high molecular weight hydrocarbon fuels. This allows about two orders of magnitude CPU time savings for multi-dimensional CFD simulations, and up to three orders of magnitude savings for the largest hydrocarbon fuel oxidation mechanisms currently available in the literature. This also sets the path for further studies. As the computational cost due to the solution of the linear system associated with the chemistry Jacobian is still the most computationally demanding task, provided that advanced numerics are adopted to compute both the Jacobian and the r.h.s. of the system of equations (sparse algebra, analytical formulation, optimal-degree interpolation of expensive functions), it is possible that further improvements could be achieved by improving the analytical representation of the Jacobian's LU factorization, or, when coupled with CFD codes, by removing the operator splitting simplification and

solve for both transport and chemistry using the same method. The issue of reaction mechanism size, that is large for dense algebra, but yet too small for extended parallelism, must be addressed to allow the integration of chemical kinetics problems on massively parallel architectures such as GPUs to be really efficient. Possibly an approach similar to CFD is needed, where every reactor is solved onto one single graphical unit, and shared memory provides a common pool for tabulated thermodynamic functions.

6. Acknowledgements

The authors gratefully acknowledge funding by the Sandia National Laboratories and the Princeton Combustion Energy Frontier Research Center (CEFRC).

7. Notes

The `SpeedCHEM` library is available upon request to the authors.

[1] R. D. Reitz, Directions in internal combustion engine research, Combustion and Flame 160 (1) (2013) 1 – 8. doi:http://dx.doi.org/10.1016/j.combustflame.2012.11.002.
URL http://www.sciencedirect.com/science/article/pii/S001021801200315X

[2] 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. doi:http://dx.doi.org/10.1016/j.pecs.2008.10.002.
URL http://www.sciencedirect.com/science/article/pii/S036012850800066X

[3] W. J. Pitz, C. J. Mueller, Recent progress in the development of diesel surrogate fuels, Progress in Energy and Combustion Science 37 (3) (2011) 330 – 350. doi:http://dx.doi.org/10.1016/j.pecs.2010.06.004.
URL http://www.sciencedirect.com/science/article/pii/S0360128510000535

[4] W. H. Green, Predictive Kinetics: A New Approach for the 21st Century, in: G. B. Marin (Ed.), Chemical Engineering Kinetics, Vol. 32 of Advances in Chemical Engineering, Academic Press, 2007, pp. 1 – 313. doi:http://dx.doi.org/10.1016/S0065-2377(07)32001-2.
URL http://www.sciencedirect.com/science/article/pii/S0065237707320012

[5] E. Hairer, S. P. Norsett, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, 2nd Edition, Springer, Berlin, 1991.

[6] S. Lam, D. Coussis, Understanding complex chemical kinetics with computational singular perturbation, Symposium (International) on Combustion 22 (1) (1989) 931 – 941. doi:http://dx.doi.org/10.1016/S0082-0784(89)80102-X.
URL http://www.sciencedirect.com/science/article/pii/S008207848980102X

[7] S. Vajda, P. Valko, T. Turanyi, Principal component analysis of kinetic models, International Journal of Chemical Kinetics 17 (1) (1985) 55–81. doi:10.1002/kin.550170107.
URL http://dx.doi.org/10.1002/kin.550170107

[8] N. J. Brown, G. Li, M. L. Koszykowski, Mechanism reduction via principal component analysis, International Journal of Chemical Kinetics 29 (6) (1997) 393–414.

[9] M. Valorani, D. A. Goussis, Explicit Time-Scale Splitting Algorithm for Stiff Problems: Auto-ignition of Gaseous Mixtures behind a Steady Shock, Journal of Computational Physics 169 (1) (2001) 44 – 79. doi:http://dx.doi.org/10.1006/jcph.2001.6709.
URL http://www.sciencedirect.com/science/article/pii/S0021999101967099

[10] U. Maas, S. Pope, Simplifying chemical kinetics: Intrinsic low-dimensional manifolds in composition space, Combustion and Flame 88 (34) (1992) 239 – 264. doi:http://dx.doi.org/10.1016/0010-2180(92)90034-M.
URL http://www.sciencedirect.com/science/article/pii/001021809290034M

[11] J. C. Keck, Rate-controlled constrained-equilibrium theory of chemical reactions in complex systems, Progress in Energy and Combustion Science 16 (2) (1990) 125 – 154. doi:http://dx.doi.org/10.1016/0360-1285(90)90046-6.
URL http://www.sciencedirect.com/science/article/pii/0360128590900466

[12] G. Li, H. Rabitz, A general analysis of exact lumping in chemical kinetics, Chemical Engineering Science 44 (6) (1989) 1413 – 1430. doi:http://dx.doi.org/10.1016/0009-2509(89)85014-6.
URL http://www.sciencedirect.com/science/article/pii/0009250989850146

[13] T. Lu, C. K. Law, A directed relation graph method for mechanism reduction, Proceedings of the Combustion Institute 30 (1) (2005) 1333 – 1341. doi:http://dx.doi.org/10.1016/j.proci.2004.08.145.
URL http://www.sciencedirect.com/science/article/pii/S0082078404001973

[14] P. Pepiot-Desjardins, H. Pitsch, An efficient error-propagation-based reduction method for large chemical kinetic mechanisms, Combustion and Flame 154 (12) (2008) 67 – 81. doi:http://dx.doi.org/10.1016/j.combustflame.2007.10.020.
URL http://www.sciencedirect.com/science/article/pii/S0010218007003264

[15] L. Elliott, D. B. Ingham, A. G. Kyne, N. S. Mera, M. Pourkashanian, C. W. Wilson, Reaction Mechanism Reduction and Optimization Using Genetic Algorithms, Industrial and Engineering Chemistry Research 44 (4) (2005) 658–667. arXiv:http://pubs.acs.org/doi/pdf/10.1021/ie049409d, doi:10.1021/ie049409d.
URL http://pubs.acs.org/doi/abs/10.1021/ie049409d

[16] F. Perini, J. L. Brakora, R. D. Reitz, G. Cantore, Development of reduced and optimized reaction mechanisms based on genetic algorithms and element flux analysis, Combustion and Flame 159 (1) (2012) 103 – 119. doi:http://dx.doi.org/10.1016/j.combustflame.2011.06.012.
URL http://www.sciencedirect.com/science/article/pii/S0010218011001921

[17] S. Pope, Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation, Combustion Theory and Modelling 1 (1) (1997) 41–63. arXiv:http://www.tandfonline.com/doi/pdf/10.1080/713665229, doi:10.1080/713665229.
URL http://www.tandfonline.com/doi/abs/10.1080/713665229

[18] 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. arXiv:http://pubs.acs.org/doi/pdf/10.1021/ef300747n, doi:10.1021/ef300747n.
URL http://pubs.acs.org/doi/abs/10.1021/ef300747n

[19] D. A. Schwer, J. E. Tolsma, W. H. Green, P. I. Barton, On upgrading the numerics in combustion chemistry codes, Combustion and Flame 128 (3) (2002) 270 – 291. doi:http://dx.doi.org/10.1016/S0010-2180(01)00352-2.
URL http://www.sciencedirect.com/science/article/pii/S0010218001003522

[20] R. A. Whitesides, M. J. McNenly, D. L. Flowers, Optimizing Time Integration of Chemical-Kinetic Networks for Speed and Accuracy, in: 8th US National Combustion Meeting, 2013.

[21] K. V. Puduppakkam, L. Liang, C. V. Naik, E. Meeks, S. L. Kokjohn, R. D. Reitz, Use of Detailed Kinetics and Advanced Chemistry-Solution Techniques in CFD to Investigate Dual-Fuel Engine Concepts, SAE Int. J. Engines 4 (2011) 1127–1149. doi:10.4271/2011-01-0895.
URL http://dx.doi.org/10.4271/2011-01-0895

[22] G. B. Ferraris, D. Manca, BzzOde: a new C++ class for the solution of stiff and non-stiff ordinary differential equation systems, Computers & Chemical Engineering 22 (11) (1998) 1595 – 1621. doi:http://dx.doi.org/10.1016/S0098-1354(98)00233-6.
URL http://www.sciencedirect.com/science/article/pii/S0098135498002336

[23] V. Damian, A. Sandu, M. Damian, F. Potra, G. R. Carmichael, The kinetic preprocessor KPP-a software environment for solving chemical kinetics, Computers and Chemical Engineering 26 (11) (2002) 1567 – 1579. doi:http://dx.doi.org/10.1016/S0098-1354(02)00128-X.
URL http://www.sciencedirect.com/science/article/pii/S009813540200128X

[24] A. Sandu, J. Verwer, M. V. Loon, G. Carmichael, F. Potra, D. Dabdub, J. Seinfeld, Benchmarking stiff ode solvers for atmospheric chemistry problems-I. implicit vs explicit, Atmospheric Environment 31 (19) (1997) 3151 – 3166. doi:http://dx.doi.org/10.1016/S1352-2310(97)00059-9.
URL http://www.sciencedirect.com/science/article/pii/S1352231097000599

[25] A. Sandu, J. Verwer, J. Blom, E. Spee, G. Carmichael, F. Potra, Benchmarking stiff ode solvers for atmospheric chemistry problems II: Rosenbrock solvers, Atmospheric Environment 31 (20) (1997) 3459 – 3472. doi:http://dx.doi.org/10.1016/S1352-2310(97)83212-8.
URL http://www.sciencedirect.com/science/article/pii/S1352231097832128

[26] P. Brown, G. Byrne, A. Hindmarsh, VODE: A Variable-Coefficient ODE Solver, SIAM Journal on Scientific and Statistical Computing 10 (5) (1989) 1038–1051. arXiv:http://epubs.siam.org/doi/pdf/10.1137/0910062, doi:10.1137/0910062.
URL http://epubs.siam.org/doi/abs/10.1137/0910062

[27] A. C. Hindmarsh, LSODE and LSODI, two new initial value ordinary differnetial equation solvers, SIGNUM Newsletter 15 (4) (1980) 10–11. doi:10.1145/1218052.1218054.
URL http://doi.acm.org/10.1145/1218052.1218054

[28] L. R. Petzold, Description of DASSL: a differential/algebraic system solver, Tech. rep., Sandia National Laboratories SAND-82-8637 (1982).

[29] M. J. McNenly, R. A. Whitesides, D. L. Flowers, Adaptive Preconditioning Strategies for Integrating Large Kinetic Mechanisms, in: 8th US National Combustion Meeting, 2013.

[30] P. Tranquilli, A. Sandu, Rosenbrock-Krylov Methods for Large Systems of Differential Equations, ArXiv 2013arXiv:1305.5481.

[31] F. Bisetti, Integration of large chemical kinetic mechanisms via exponential methods with Krylov approximations to Jacobian matrix functions, Combustion Theory and Modelling 16 (3) (2012) 387–418. arXiv:http://www.tandfonline.com/doi/pdf/10.1080/13647830.2011.631032, doi:10.1080/13647830.2011.631032.
URL http://www.tandfonline.com/doi/abs/10.1080/13647830.2011.631032

[32] Y. Shi, W. H. Green, H.-W. Wong, O. O. Oluwole, Redesigning combustion modeling algorithms for the Graphics Processing Unit (GPU): Chemical kinetic rate evaluation and ordinary differential equation integration, Combustion and Flame 158 (5) (2011) 836 – 847. doi:http://dx.doi.org/10.1016/j.combustflame.2011.01.024.
URL http://www.sciencedirect.com/science/article/pii/S0010218011000393

[33] J. C. Linford, J. Michalakes, M. Vachharajani, A. Sandu, Multi-core acceleration of chemical kinetics for simulation and prediction, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, ACM, New York, NY, USA, 2009, pp. 7:1–7:11. doi:10.1145/1654059.1654067.
URL http://doi.acm.org/10.1145/1654059.1654067

[34] O. Herbinet, W. J. Pitz, C. K. Westbrook, Detailed chemical kinetic oxidation mechanism for a biodiesel surrogate, Combustion and Flame 154 (3) (2008) 507 – 528. doi:http://dx.doi.org/10.1016/j.combustflame.2008.03.003.
URL http://www.sciencedirect.com/science/article/pii/S0010218008000771

[35] F. Perini, E. Galligani, G. Cantore, R. Reitz, Validation of a Sparse Analytical Jacobian Chemistry Solver for Heavy-Duty Diesel Engine Simulations with Comprehensive Reaction Mechanisms, SAE Technical Paper 2012-01-1974doi:10.4271/2012-01-1974.
URL http://dx.doi.org/10.4271/2012-01-1974

[36] F. Perini, A. Krishnasamy, Y. Ra, R. D. Reitz, Computationally Efficient Simulation of Multi-component Fuel Combustion using a Sparse Analytical Jacobian Chemistry Solver and High-dimensional Clustering, in: Proceedings of the ASME Internal Combustion Engine Division's 2013 Fall Technical Conference ICEF2013-19039, 2013.

[37] R. S. Varga, Factorization and normalized iterative methods, in: R. E. Langer (Ed.), Boundary Problems in Differential Equations, University of Wisconsin Press, 1960, pp. 121–142.

[38] Y. Saad, ILUT: A dual threshold incomplete LU factorization, Numerical Linear Algebra with Applications 1 (4) (1994) 387–402. doi:10.1002/nla.1680010405.
URL http://dx.doi.org/10.1002/nla.1680010405

[39] A. Patel, S.-C. Kong, R. D. Reitz, Development and Validation of a Reduced Reaction Mechanism for HCCI Engine Simulations, SAE Technical Paper 2004-01-0558doi:10.4271/2004-01-0558.
URL http://dx.doi.org/10.4271/2004-01-0558

[40] Y. Ra, R. D. Reitz, A reduced chemical kinetic model for IC engine combustion simulations with primary reference fuels, Combustion and Flame 155 (4) (2008) 713 – 738. doi:http://dx.doi.org/10.1016/j.combustflame.2008.05.002.
URL http://www.sciencedirect.com/science/article/pii/S0010218008001351

[41] H. Wang, PRF reaction mechanism, personal communication (2013).

[42] Y. Ra, R. D. Reitz, A combustion model for IC engine combustion simulations with multi-component fuels, Combustion and Flame 158 (1) (2011) 69 – 90. doi:http://dx.doi.org/10.1016/j.combustflame.2010.07.019.
URL http://www.sciencedirect.com/science/article/pii/S0010218010002075

[43] R. Seiser, H. Pitsch, K. Seshadri, W. Pitz, H. Curran, Extinction and autoignition of n-heptane in counterflow configuration, Proceedings of the Combustion Institute 28 (2) (2000) 2029 – 2037. doi:http://dx.doi.org/10.1016/S0082-0784(00)80610-4.
URL http://www.sciencedirect.com/science/article/pii/S0082078400806104

[44] H. Curran, P. Gaffuri, W. Pitz, C. Westbrook, A comprehensive modeling study of iso-octane oxidation, Combustion and Flame 129 (3) (2002) 253 – 280. doi:http://dx.doi.org/10.1016/S0010-2180(01)00373-X.
URL http://www.sciencedirect.com/science/article/pii/S001021800100373X

[45] C. K. Westbrook, W. J. Pitz, O. Herbinet, H. J. Curran, E. J. Silke, A comprehensive detailed chemical kinetic reaction mechanism for combustion of n-alkane hydrocarbons from n-octane to n-hexadecane, Combustion and Flame 156 (1) (2009) 181 – 199. doi:http://dx.doi.org/10.1016/j.combustflame.2008.07.014.
URL http://www.sciencedirect.com/science/article/pii/S0010218008002125

[46] R. Kee, F. M. Rupley, A. Miller, Chemkin-II: A Fortran chemical kinetics package for the analysis of gas-phase chemical kinetics, Tech. rep., Sandia National Laboratories SAND-89-8009 (1989).

[47] Y. Saad, Iterative Methods for Sparse Linear Systems, EngineeringPro collection, Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2003.
URL http://books.google.com/books?id=h9nwszYPblEC

[48] J. Reid, The new features of Fortran 2008, SIGPLAN Fortran Forum 27 (2) (2008) 8–21. doi:10.1145/1408643.1408645.
URL http://doi.acm.org/10.1145/1408643.1408645

[49] R. M. Stallman, et al., Gnu compiler collection (November 1987).
URL http://gcc.gnu.org

[50] A. Amsden, P. Orourke, T. Butler, KIVA-2: A computer program for chemically reactive flows with sprays, NASA STI/Recon Technical Report N 89 (1989) 27975.

[51] M. J. Holst, Notes on the kiva-ii software and chemically reactive fluid mechanics, Tech. rep., Lawrence Livermore National Laboratory (1992).

[52] A. A. Amsden, KIVA-3V: A block-structured KIVA program for engines with vertical or canted valves, Tech. rep., Los Alamos National Lab., NM (United States) (1997).

[53] D. J. Torres, M. F. Trujillo, KIVA-4: An unstructured ALE code for compressible gas flow with sprays, Journal of Computational Physics 219 (2) (2006) 943 – 975. doi:http://dx.doi.org/10.1016/j.jcp.2006.07.006.
URL http://www.sciencedirect.com/science/article/pii/S002199910600338X

[54] H. Eyring, The Activated Complex in Chemical Reactions, The Journal of Chemical Physics 3 (2) (1935) 107–115. doi:10.1063/1.1749604.
URL http://link.aip.org/link/?JCP/3/107/1

[55] A. A. Konnov, Remaining uncertainties in the kinetic mechanism of hydrogen combustion, Combustion and Flame 152 (4) (2008) 507 – 528. doi:http://dx.doi.org/10.1016/j.combustflame.2007.10.024.
URL http://www.sciencedirect.com/science/article/pii/S0010218007003318

[56] I. Duff, A. Erisman, J. Reid, Direct Methods for Sparse Matrices, Monographs on numerical analysis, Oxford University Press, Incorporated, 1989.

[57] E. Cuthill, J. McKee, Reducing the bandwidth of sparse symmetric matrices, in: Proceedings of the 1969 24th national conference, ACM '69, ACM, New York, NY, USA, 1969, pp. 157–172. doi:10.1145/800195.805928.
URL http://doi.acm.org/10.1145/800195.805928

[58] A. George, J. W. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice Hall Professional Technical Reference, 1981.

[59] R. Tewarson, Sparse Matrices, Mathematics in Science and Engineering, Academic Press, 1973.

[60] J. Troe, Fall-off Curves of Unimolecular Reactions, Berichte der Bunsengesellschaft fur physikalische Chemie 78 (5) (1974) 478–488. doi:10.1002/bbpc.19740780510.
URL http://dx.doi.org/10.1002/bbpc.19740780510

[61] C. Kelley, Iterative Methods for Linear and Nonlinear Equations, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, 1995.
URL http://books.google.com/books?id=c3m7Ro7EhOAC

[62] J. Ortega, Numerical Analysis: A Second Course, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 1990.
URL http://books.google.com/books?id=DboVs3T2UnoC

[63] Y. Saad, M. Schultz, GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, SIAM Journal on Scientific and Statistical Computing 7 (3) (1986) 856–869. arXiv:http://epubs.siam.org/doi/pdf/10.1137/0907058, doi:10.1137/0907058.
URL http://epubs.siam.org/doi/abs/10.1137/0907058

[64] C. Gear, Numerical initial value problems in ordinary differential equations, Prentice-Hall series in automatic computation, Prentice-Hall, 1971.
URL http://books.google.com/books?id=e9QQAQAAIAAJ

[65] P. N. Brown, A. C. Hindmarsh, Matrix-free methods for stiff systems of ODE's, SIAM J. Numer. Anal. 23 (3) (1986) 610–638. doi:10.1137/0723039.
URL http://dx.doi.org/10.1137/0723039

[66] R. Dembo, S. Eisenstat, T. Steihaug, Inexact Newton Methods, SIAM Journal on Numerical Analysis 19 (2) (1982) 400–408. arXiv:http://epubs.siam.org/doi/pdf/10.1137/0719025, doi:10.1137/0719025.
URL http://epubs.siam.org/doi/abs/10.1137/0719025

[67] S. C. Eisenstat, H. F. Walker, Choosing the Forcing Terms in an Inexact Newton Method, SIAM J. Sci. Comput 17 (1994) 16–32.

[68] W. Rheinboldt, Methods for Solving Systems of Nonlinear Equations, CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, 1998.
URL http://books.google.com/books?id=lnJznmVnJ2wC

[69] W. E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Quart. Appl. Math 9 (1) (1951) 17–29.

[70] I. C. Ipsen, C. D. Meyer, The Idea Behind Krylov Methods, The American Mathematical Monthly 105 (1998) 889–899.

[71] A. Greenbaum, Iterative Methods for Solving Linear Systems, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, 1997.
URL http://books.google.com/books?id=WwMDNLxrwocC

[72] S.-L. Kim, J.-Y. Choi, I.-S. Jeung, Y.-H. Park, Application of approximate chemical Jacobians for constant volume reaction and shock-induced combustion, Applied Numerical Mathematics 39 (1) (2001) 87 – 104. doi:http://dx.doi.org/10.1016/S0168-9274(01)00054-X.
URL http://www.sciencedirect.com/science/article/pii/S016892740100054X

[73] G. D. Byrne, Pragmatic experiments with Krylov methods in the stiff ODE settings, in: Computational Ordinary Differential Equations (J.R. Cash and I. Gladwell, eds.), Clarendon Press, Oxford, UK, 1992.

[74] P. Brown, A. Hindmarsh, L. Petzold, Using Krylov Methods in the Solution of Large-Scale Differential-Algebraic Systems, SIAM Journal on Scientific Computing 15 (6) (1994) 1467–1488. arXiv:http://epubs.siam.org/doi/pdf/10.1137/0915088, doi:10.1137/0915088.
URL http://epubs.siam.org/doi/abs/10.1137/0915088

[75] F. Perini, High-dimensional, unsupervised cell clustering for computationally efficient engine simulations with detailed combustion chemistry, Fuel 106 (2013) 344 – 356. doi:http://dx.doi.org/10.1016/j.fuel.2012.11.015.
URL http://www.sciencedirect.com/science/article/pii/S0016236112008915

[76] I. W. Ekoto, W. F. Colban, P. C. Miles, S. W. Park, D. E. Foster, R. D. Reitz, U. Aronsson, O. Andersson, UHC and CO Emissions Sources from a Light-Duty Diesel Engine Undergoing Dilution-Controlled Low-Temperature Combustion, SAE International Journal of Engines 2 (2) (2010) 411–430. arXiv:http://saeeng.saejournals.org/content/2/2/411.full.pdf+html, doi:10.4271/2009-24-0043.
URL http://saeeng.saejournals.org/content/2/2/411.abstract

[77] F. Perini, A. Dempsey, R. Reitz, D. Sahoo, B. Petersen, P. Miles, A Computational Investigation of the Effects of Swirl Ratio and Injection Pressure on Mixture Preparation and Wall Heat Transfer in a Light-Duty Diesel Engine, SAE Technical Paper 2013-01-1105doi:10.4271/2013-01-1105.
URL http://dx.doi.org/10.4271/2013-01-1105