Journal of Computational and Applied Mathematics 226 (2009) 197–204

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A distributed combustion solver for engine simulations on grids

Laura Antonelli [a], Paola Belardini [b], Pasqua D'Ambra [a,*], Francesco Gregoretti [a], Gennaro Oliva [a]

[a] *Institute for High-Performance Computing and Networking, CNR, Via P. Castellino 111, I-80131 Naples, Italy*

[b] *Engine Institute, CNR, Via G. Marconi 8, I-80125 Naples, Italy*

ARTICLE INFO

MSC:
65L05
68U20
80A25

Keywords:
Stiff ODE solvers
Adaptive algorithms
Detailed combustion models
Engine simulations
Grid computing

ABSTRACT

Multi-dimensional models for predictive simulations of modern engines are an example of multi-physics and multi-scale mathematical models, since lots of thermofluiddynamic processes in complex geometrical configurations have to be considered. Typical models involve different submodels, including turbulence, spray and combustion models, with different characteristic time scales. The predictive capability of the complete models depends on the accuracy of the submodels as well as on the reliability of the numerical solution algorithms. In this work we propose a multi-solver approach for reliable and efficient solution of the stiff Ordinary Differential Equation (ODE) systems arising from detailed chemical reaction mechanisms for combustion modeling. Main aim was to obtain high-performance parallel solution of combustion submodels in the overall procedure for simulation of engines on distributed heterogeneous computing platforms. To this aim we interfaced our solver with the CHEMKIN-II package and the KIVA3V-II code and carried out multi-computer simulations of realistic engines. Numerical experiments devoted to test reliability of the simulation results and efficiency of the distributed combustion solver are presented and discussed.

1. Introduction

The design of modern engines relies on new technologies devoted to enhance fuel conversion efficiency and reduce pollutant emissions, in order to match the stringent limits on NO_x, CO_x and soot formation imposed by the Governments. The impact on computational modeling is the need of accurately describe highly complex, different physical–chemical phenomena occurring in each engine cycle, and of reliably perform over a wide range of engine operating conditions. Mathematical models for the description of the overall problem typically involve unsteady Navier–Stokes equations for turbulent multi-component mixtures of ideal gases, coupled with suitable equations for fuel spray and combustion. The solution of the overall model usually relies on an operator splitting technique, where the different physical phenomena are decoupled, and different submodels are separately solved on a 3D computational grid.

In recent years much attention has been addressed to the combustion submodels, by introducing detailed chemical reaction mechanisms, where the number of the chemical species and the reactions to be considered reach also several hundreds. The main computational kernel in this framework is the solution of stiff systems of non-linear ODEs for which implicit methods have to be employed. Therefore, the numerical solution of chemistry has become one of the most computationally demanding parts in simulations, thus leading to the need of efficient combustion solvers [13–16]. In this work we propose a new approach for solving the above systems based on a combination of two different implicit solvers:

* Corresponding author.
E-mail addresses: laura.antonelli@na.icar.cnr.it (L. Antonelli), p.belardini@im.cnr.it (P. Belardini), pasqua.dambra@na.icar.cnr.it (P. D'Ambra), francesco.gregoretti@na.icar.cnr.it (F. Gregoretti), gennaro.oliva@na.icar.cnr.it (G. Oliva).

0377-0427/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2008.08.002

VODE [8] and SDIRK4 [12], in order to balance the accuracy and efficiency requirements in realistic simulations. Our approach is based on the information that some species, such as ketones and OH radical intermediate species with low density and short characteristic time scales, are crucial in the so-called *cold combustion phase*, therefore stiff-stability and accuracy of the Backward Differentiation Formulas (BDF) implemented in VODE are the method of choice. On the other hand, in the so-called *hot combustion phase*, ketones can be rapidly damped while other low-density species such as OH radical remains crucial. In this phase, accuracy remains important but highly oscillating components can be observed, for which *L*-stability properties of the Singly Diagonally Implicit Runge–Kutta (SDIRK) method implemented in the SDIRK4 package appears more effective for having reliable solution and saving computing time. On the other hand, within the context of a time-splitting technique, where combustion is decoupled by fluid flow, the chemical reactions do not introduce any coupling among grid cells, therefore combustion models show an intrinsic parallelism to be exploited for making possible ever more detailed chemical models and advanced solution methods, as reported in [1,4,16]. In this work we also focus on the efficient implementation of the combustion solver in distributed heterogeneous environments, by using MJMS, a Multisite Job Management System for execution of MPI applications in a Grid environment [9]. The combustion solver is based on the CHEMKIN-II package for managing detailed chemistry and it is interfaced with the sequential KIVA3V-II code [2] for the simulation of the entire engine cycle. The paper is organized as follows. In Section 2 we report the mathematical description of the combustion model. In Section 3 we describe the main features of the ODE solvers we used, both in terms of employed formulas and in terms of stability properties. In Section 4 we outline the software architecture in which our solver has been integrated for distributed multisite engine simulations. In Section 5 we discuss simulation results for a commercial automotive engine, running with three different operating conditions. Some conclusions and plans for future work are included in Section 6.

2. Combustion model for diesel engines

Diesel engine combustion is characterized by liquid fuel injection in a turbulent environment. Two main phases can be distinguished in the overall phenomenon. Fuel injection gives raise to chemical reactions that, under suitable temperature and pressure conditions, lead to fuel ignition. The period from the starting of fuel injection and fuel ignition is named *ignition delay*: in this phase chemical reactions occur without giving strong energy contributions, but a high stiffness degree is the main feature, due to very different reaction rates among the reactant species. Kinetics occurring before ignition is usually referred to as low-temperature combustion or *cold combustion*, while *hot combustion phase*, or high-temperature combustion, is the chain of reactions subsequent to ignition.

We consider a combustion model based on a modification of a detailed kinetic scheme introduced in [11]. It considers N-dodecane as primary fuel and involves 62 chemical species and 285 reactions. The kinetic scheme describes the H abstraction and the oxidation of the primary fuel, with production of alchil-peroxy-radicals, followed by the ketoydroperoxide branching. In the model the fuel pyrolysis determines the ketones and olefins formation. Moreover, a scheme of soot formation and oxidation is provided, together with a classical scheme of NO_x formation. The reaction system is expressed by the following system of non-linear ODEs:

$$\dot{\rho}_m = W_m \sum_{r=1}^{R} (b_{mr} - a_{mr}) \dot{\omega}_r(\rho_1, \ldots, \rho_m, T), \quad m = 1, \ldots, M, \tag{1}$$

where R is the number of chemical reactions involved in the system, $\dot{\rho}_m$ is the production rate of species m, W_m is its molecular weight, a_{mr} and b_{mr} are integral stoichiometric coefficients for reaction r and $\dot{\omega}_r$ is the kinetic reaction rate. Production rate terms can be separated into creation rates and destruction rates [13]:

$$\dot{\rho}_m = \dot{C}_m - \dot{D}_m, \quad m = 1, \ldots M, \tag{2}$$

where $\dot{C}_m$, $\dot{D}_m$ are the creation and the destruction rate of species m, respectively. The latter can be expressed as

$$\dot{D}_m = \frac{X_m}{\tau_m}, \quad m = 1, \ldots M, \tag{3}$$

where X_m, τ_m are the molar concentration and the characteristic time for destruction rate of species m, respectively. Eq. (3) shows that the eigenvalues of the Jacobian matrix of the right-hand side of system (1) are related to the characteristic times for destruction rates of species involved in the combustion model. Our detailed reaction model involves a great number of intermediate species and no equilibrium assumption is made, therefore high stiffness degrees characterize the ODE systems. Indeed by a deep analysis of the characteristic times for the destruction rates, as estimated by the CHEMKIN library routines within a typical engine simulation when VODE solver is used, we can see that they vary in the interval $[\approx 0, 10^{18}]$ during the cold combustion phase. On the other hand, highly oscillating components, corresponding to complex eigenvalues with a large imaginary part, can be observed in the ODE systems to be solved during the hot combustion phase.

3. ODE solvers and multi-solver approach

In this Section we briefly describe main features of the general-purpose ODE solvers we used for combustion model. In order to motivate our choices, we recall some concepts that play a special role in the framework of the solution of stiff systems. For a deep insight the matter we send the reader to scientific literature about the topic.

In the following, we refer to a well-posed and stable (no eigenvalue of the Jacobian matrix has a positive real part) initial value ordinary differential problem expressed in the general form:

$$\begin{cases} \dot{y} = f(t, y) \\ y(t_0) = y_0 \end{cases} \tag{4}$$

with $t \in I \subseteq \Re$, $y \in \Re^N$. The problem (4) is called *stiff* if at least one eigenvalue of the Jacobian matrix has large absolute value of real part.

Let S be the region of linear stability and $R(z)$ the associate, complex, linear stability function of an ODE discretization method. We start recalling that a method is *absolutely stable* (*A-stable*) if $S \supseteq \{z \in C : \mathrm{Re}(z) \leq 0\}$. It is well known [3,12] that absolute stability does not guarantee efficient solution of stiff problems; *ad hoc* properties, such as, stiff and L-stability have been stated in order to characterize methods that are appropriate in this framework. We consider $R_1 = \{z \in C : \mathrm{Re}(z) \leq -a\}$ and $R_2 = \{z \in C : -a \leq \mathrm{Re}(z) < 0, -c \leq \mathrm{Im}(z) \leq c\}$ with $a, c > 0$. If $S \supseteq R_1 \cup R_2$ the method is said *stiffly stable*. Stiff stability was introduced by Gear [10] in his analysis of linear multi-step methods. It is a weaker property than absolute stability, since it requires that the method is stable in a subplane of the complex plane C when dealing with large real part complex numbers, while it relaxes the required conditions close to the origin. *L-stability* is instead stronger than *A*-stability: if the method is *A*-stable and $\lim_{z\to\infty} R(z) = 0$, the method is said *L*-stable. The latter property is motivated by the fact that *A*-stability can not preserve a method from producing values, in successive iterations, that have the same magnitudo order; this, obviously, is not desirable in the solution of very stiff problems.

VODE solver is a well-known software designed for both stiff and non-stiff systems. It is widely used together with the CHEMKIN package for stiff systems arising from chemical reaction modeling. In the stiff case, it uses variable coefficient stiffly stable Backward Differentiation Formulas (BDF) [12]:

$$\sum_{i=0}^{q} \alpha_{n,i} y_{n-i} + h_n \beta_n \dot{y}_n = 0, \tag{5}$$

of orders q from one through five, with an automatic, adaptive technique for the selection of step sizes. VODE provides several methods for the solution of the systems of algebraic equations; in our simulations we used modified Newton method with numerically computed Jacobian and we set the VODE option for Jacobian reusing in order to preserve efficiency. In a first approach, we used VODE for each system of a computational grid representing the engine, with local automatic choice of the time steps per each system. Typically, a formula of high order (usually five) is used by the solver during the integration at each time-splitting interval and this order is kept constant until convergence is reached within the fixed value for internal time steps. Two cases of failures of VODE have been observed in our experiments: in one case, due to repeated reductions of integration internal time steps, the solver suggests to increase the maximum number of internal iterations in order to get convergence within accuracy requests; while, in the other case, due to repeated error test failures for one attempted time step, the solver suggests to continue the integration task by substituting the used formula with the one order formula, that is the *L*-stable backward Euler method. Taking into account the above information, we applied VODE by using the following strategy:

Increasing iterations. Our model includes intermediate species, such as ketones and OH radical, with very low density, whose accurate solution is needed to well understand the combustion process and reliably identify the starting of ignition. In order to get accurate solution of the systems, when the maximum number (fixed to 2500) of internal iterations has been reached before completing the integration task on the splitting interval, a new call to VODE is done. The last value of the solution is used as starting point and the integration is attempted on the rest of the current interval within other 2500 internal iterations. In case of failure after the second VODE call, the integration task on the rest of the current interval is attempted, within no more than 2500 internal iterations, by using the one-order formula. This choice is related to possible system components with eigenvalues having large imaginary part.

Changing formula. In the case of repeated error test failures for one attempted time step, we consider the problem due to highly oscillating components, therefore the integration task on the rest of the current interval is attempted by using the one-order formula, as also suggested by VODE error messages. Also in this case, the maximum number for internal iterations is set to 2500.

Numerical results on our test cases related to the use of the above strategy both in terms of reliability and in terms of efficiency are discussed in Section 5.

Since after ignition highly oscillating components should be rapidly damped in order to well simulate the hot (smooth) combustion phase (see also [14]), we proposed a multi-solver strategy for the solution of the chemistry systems in engine

simulations, whose first results have been presented in [6]. Our idea is based on the use of the stiff solver VODE, as in the above explained strategy, for the solution of the chemistry systems in the cold combustion phase of engine simulations, and in the use of the SDIRK4 solver for the solution of all the chemistry systems in the hot combustion phase. The switching criterion between the two phases is based on a physical evidence, that is the OH radical can be considered a reliable hot combustion marker, indeed its total mass in the combustion chamber becomes meaningful only after ignition.

SDIRK4 is based on a 5-stages SDIRK method of order four, with variable step size control. An s-stages Runge–Kutta (RK) method can be expressed in the general form

$$\begin{cases} Y_i = y_{n-1} + h_n \sum_{j=1}^{s} a_{ij} f(t_{n-1} + c_j h_n, Y_j), \quad i = 1, s \\ y_n = y_{n-1} + h_n \sum_{i=1}^{s} b_i f(t_{n-1} + c_i h_n, Y_i), \end{cases} \tag{6}$$

that is, any particular RK method is characterized by a special choice of matrix $A = (a_{ij})$ and vector $b = (b_i)$, and $c_i = \sum_{j=1}^{s} a_{ij}$, $i = 1, \ldots, s$. A RK method is said singly diagonally implicit if A is lower triangular and $a_{ii} = a \; \forall i = 1, s$. SDIRK method implemented in SDIRK4 is more attractive than high-order BDF for the solution of highly oscillating stiff systems since it is L-stable [12]. The main computational kernel in this framework is the solution of s linear systems of dimension $N \times N$, all having the same coefficient matrix, at each time step; thus, only one Jacobian evaluation and one LU factorization is required at each time step. In our experiments, we considered the default choice for parameter setting and the software option for a numerical internally computed full Jacobian. Note that a discussion on the use of SDIRK4 in the overall engine simulation process and on its poor results in ignition delay prediction is reported in [5].

4. Distributed solution of combustion

In this Section we describe the main features of the software component we developed for distributed solution of chemical reaction schemes in simulation of diesel engines. Since reaction schemes do not introduce any coupling among the grid cells representing the combustion chamber, the solution of the ODE systems is a so-called inherently distributed problem, therefore we can exploit modern features of Grid environments in order to obtain high-performance solution of reaction schemes in large-scale engine simulations.

Note that physical stiffness is strongly related to local conditions, therefore, when adaptive solvers are considered in a distributed environment, also including heterogeneous resources, data partitioning and process allocation become critical issues for computational load balancing and reduction of idle times.

In order to reduce the impact of local stiffness and adaptive solution strategies on a possible computational load imbalance, our software component supports a data distribution where systems of ODEs related to contiguous cells are assigned to different processors. To this aim, grid cells are reordered according to a permutation of indices, deduced by a pseudo-random sequence, and the ODE systems per each grid cell are distributed among the available processes, following the new order of the grid cells. Furthermore, in order to take into account possible load imbalance due to the use of heterogeneous resources, the ODE systems are distributed on the basis of CPU performances, as we explain in the following, so that faster processors get more workload.

Our software is written in Fortran 77 and it is based on CHEMKIN-II [13], a software package for managing large models of chemical reactions in the context of simulation software. It provides a database and a software library for computing model parameters involved in system (1). The parallel software component for combustion modeling is also interfaced with the sequential KIVA3V-II code, in order to properly test it within real simulations. The KIVA family of codes has been developed since '80 years at Los Alamos National Laboratories and its modifications are being used worldwide for engine applications. However, the various versions of the code include reduced chemical reaction mechanisms for combustion modeling and numerical algorithms that are inadequate for coupled solution of detailed mechanisms. For our aim, we substituted the original KIVA combustion submodel with our combustion software.

The distributed implementation of our combustion solver relies on a job management system, named MJMS (Multisite Jobs Management System) [9]. This system interacts with the pre-ws services of the Globus Toolkit 4.0 for job submission and monitoring and to get information about available Grid resources, with the MPICH-G2 implementation of MPI for process synchronization and with the Condor-G system for job management. MJMS allows the users to submit execution requests for multi-site parallel applications which consist of multiple distributed processes running on one or more potentially heterogeneous computing resources in different locations. The processes are mapped on the available resources according to requirements and preferences specified by the user in order to meet the application needs. In our context we required the combustion solver to be executed on different parallel machines that provide a total aggregate peak performance of at least 30 000 M flops. The resources are automatically selected by MJMS that also makes available the information about the number of processors for each computing resource and the corresponding CPU performances. The application takes into account the above information and configures itself at run time in order to perform a balanced load distribution. The data distribution is achieved through the algorithm described as follow. Let $p_1, \ldots, p_n$ be the number of processors,

Table 1
Multijet 16V engine characteristics

Bore (mm)	82.0
Stroke (mm)	90
Compression ratio	16.5:1
Engine speed	1500 rpm
Displacement (cm^3)	475
Valves per cylinder	4
Injection system	2nd gen. common rail
Electro-injector	Microsac, 7 holes, $\phi = 140$ mm, 440 mm^3/30 s/100 bar

respectively, for the n computing resources $c_1, \ldots, c_n$ selected by the system and let $cpow_1, \ldots, cpow_n$ be the corresponding CPU performances. The algorithm computes

$$pow_{\min} = \min\{cpow_1, \ldots., cpow_n\}$$

and $k_1, \ldots, k_n$ such that

$$k_i = \frac{cpow_i}{pow_{\min}} \quad i = 1, \ldots, n.$$

Then it computes the factor

$$f = \frac{1}{\sum_{i=1}^{n} p_i k_i}$$

so that each processor of the c_i computing resource get the fraction $f \cdot k_i$ of the total workload, i.e. each processor, at each time step, owns the data for solving the ODE systems related to $f \cdot k_i \cdot ncells$ computational grid cells, where *ncells* is the total number of active grid cells. Note that in the previous algorithm, at the moment, we neglected possible performance degradations due to network heterogeneity and interprocess communication and synchronization, since our problem is a so-called compute-intensive problem, i.e. the ratio between computation and communication is largely favorable.

5. Numerical experiments and performance results

In this Section we show results concerning engine simulations performed on a prototype, single cylinder diesel engine, having characteristics similar to the 16 valves EURO IV Fiat 1.9 JTD Multijet. Main engine parameters are reported in Table 1.

Three different operating conditions have been considered, corresponding to different values of Exhaust Gas Recirculation (EGR), rail pressure, and injection timing, which characterized three different test cases (*Case* 1–3) for our simulation software.

Our typical computational grid was a 3D cylindrical sector representing a sector of the engine cylinder and the piston bowl. It was formed by about 3000 cells, numbered in counter-clockwise fashion on each horizontal layer, from bottom-up. Note that, the structure of the active computational grid changes within each simulation of the entire engine cycle in order to follow the piston movement into the cylinder. The limit positions of the piston, that is the lowest point from which it can leave and the highest point it can reach, are expressed with respect to the so-called Crank angle values and they correspond to $-180°$ and $0°$. During the typical interval of the Crank angles ($[-20°, 40°]$) in which main combustion phenomena happen, the total number of active cells is about 1000.

ODE systems have been solved both by means the VODE solver and the multisolver, as explained in Section 3. In the stopping criteria of the solvers, both relative and absolute error control tolerances were considered; at this purpose, we defined two vectors, *rtol* and *atol*, respectively. In all the experiments here analyzed *atol* values were fixed in dependence of the particular chemical species. The reason motivating this choice relies on the very different concentrations characterizing chemical species involved in detailed reaction models. All the components of *rtol* were set to 10^{-3}, in order to satisfy the application accuracy request. At each internal time step of the ODE solvers, the so-called clipping technique was applied, i.e. negative concentrations were set equal to zero, to guarantee the positivity of the chemical species.

In the following we first discuss numerical results in terms of reliability and efficiency, by comparing sequential executions of the software component based on VODE versus the software component based on the multisolver, then we show efficiency results of distributed engine simulations for the most efficient solver. All the times are the mean values over three executions, to reduce the variability in the measurements, and they correspond to the total execution times of the KIVA3V-II code, when it is interfaced with our combustion solver, for a complete engine cycle.

We carried out our experiments using a small Grid testbed, namely *Engine Grid*, including the following two Linux clusters:

- *Vega*: a Beowulf-class cluster of 16 nodes connected via Fast Ethernet, operated by the Naples Branch of the Institute for High-Performance Computing and Networking (ICAR-CNR). Each processor is equipped with a 1.5 GHz Pentium IV processor and a RAM of 512 MB. The GNU 2.9 compiler suite is available on this cluster.

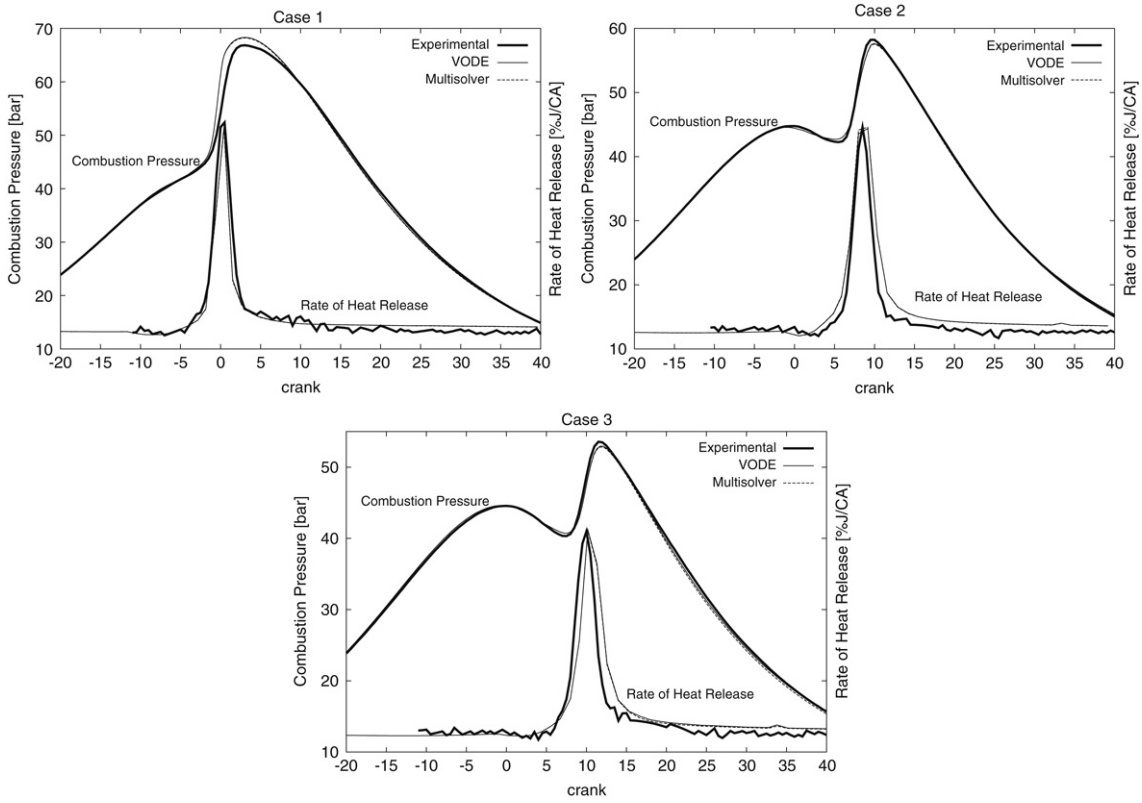

Fig. 1. Combustion pressure and heat release evolution for the three available test cases.

- *Imbeo*: a Beowulf-class cluster of 16 nodes connected via Fast Ethernet, operated by the Engine Institute (IM-CNR). Eight nodes are equipped with a 2.8 GHz Pentium IV processor, while the others have a 3.8 GHz Pentium IV processor. All the processors have a RAM of 1 GB. The GNU 3.2 compiler suite is available on this cluster.

The Globus Toolkit 4.0 and MPICH-G2 rel. 1.2.5.2 are installed on both clusters as Grid middleware; MPICH 1.2.5 based on the ch_p4 device is the local version of MPI library. The above clusters are automatically selected by MJMS from the *Engine Grid* when asking for an aggregate computing power of more than 30 000 M flops.

5.1. VODE versus multisolver

Sequential simulations of our software have been carried out on one node of the *Imbeo* cluster. In the following we compare results obtained by using the software component based on VODE with results obtained by using the multi-solver approach. In Fig. 1 we show the volume-averaged cylinder pressure and heat-release evolution on the three different test cases, obtained with the two different solution approaches. We can see that numerical solutions obtained by using VODE and the multi-solver approach are comparable. On the other hand, numerical simulations well predict ignition delay, while some differences between experimental and numerical data can be observed in the maximum values for the pressure, which in the worst case (*Case* 1) are lower than 2%, i.e. they are in the range of the measurement errors for the experimental setting. Therefore, we can say that our numerical solvers are able to reliably simulate the entire engine cycle for the three test cases, representative of very different operating conditions of the engine.

In Fig. 2 we also report the total mass evolution of the OH radical during the process for the three different test cases. Note that, by a comparison of the combustion pressure evolution and the total mass of the OH radical for all the three cases, ignition delay is well marked by a total mass greater than 10^{-5}. In our simulations with the multi-solver approach, we used a threshold of $2. \times 10^{-5}$ for the computed total mass of the OH species to switch from VODE to SDIRK4. This leads to change the ODE solver at the following Crank angles for the three test cases, respectively: $Ca = -0.46$, $Ca = 8.03$, $Ca = 9.83$. A deep experimental analysis devoted to the reliable use of the total mass of OH radical as ignition delay marker can be find in [7].

In Table 2 we show results related to the computational efficiency of the two approaches. In particular we show the total execution times (in seconds) for the simulation of a complete engine cycle and some parameters related to the ODE solvers. These are the total number of function evaluations (NFE), the total number of Jacobian evaluations (NJE) and the total number of LU factorizations (NLU) performed during the simulation. The results are summarized for the two ODE solution

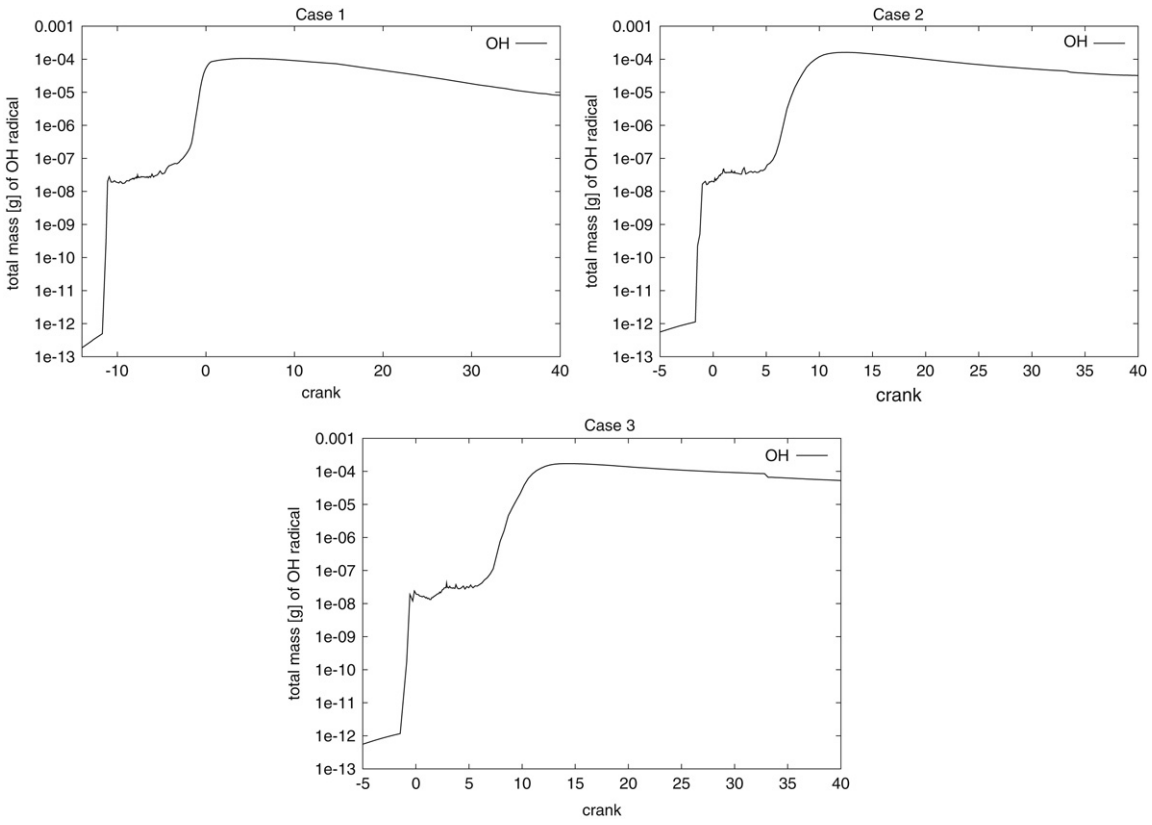

Fig. 2. OH radical total mass evolution for the three available test cases.

Table 2
Performance of the VODE solver versus the multi-solver approach for the three available test cases

	Exec. Time	NFE	NJE	NLU
	VODE			
Case 1	48 739	165 892 617	2091 449	9 449 589
Case 2	56 373	195 128 472	2932 842	11 803 695
Case 3	56 757	218 955 204	2839 449	11 214 597
	Multisolver			
Case 1	39 002	53 632 750	1270 157	5604 574
Case 2	44 591	59 091 574	1471 028	6357 436
Case 3	37 458	53 474 938	1225 802	5394 631

strategies on the three available test cases. We can see that for each test case, the best efficiency has been obtained by using the multi-solver approach. We obtained a reduction of about 20% in the total execution time for the first two cases and of about 34% for the *Case* 3, originated by a strong reduction of the total number of function and Jacobian evaluations, as well as of the LU factorizations, required by the multisolver with respect to VODE.

It is worth noting that by a monitoring of the failures of the VODE solver, when it is also used in the hot combustion phase, we observed that the backward Euler method has been used on a total of 179 cells for the *Case* 1, on a total of 183 cells for the *Case* 2, and on a total of 178 cells for the *Case* 3, corresponding to about 18% of the active cells. On the other hand, the Backward Euler has been used in the cold combustion phase a maximum of 15 cells for all the three test cases.

5.2. Performance of distributed simulations

In our engine simulations on the *Engine Grid*, we ran engine simulations for our three test cases both on each cluster, using the local version of MPI, and also on the Grid testbed via MPICH-G2. MJMS gathers the information about the CPU performances provided by its Information System, that is an extension of the Globus Information System, and the application configures itself at run time by using those information as $cpow_i$ values. The value obtained for the single processor of Vega is 534 M flops, while a value of about 1474 M flops is obtained for the processors of Imbeo. Therefore, the workload factor

Table 3
Total execution times (in seconds) of distributed simulations

	Case 1	Case 2	Case 3
Vega	8548	12 521	8629
Imbeo	5057	6 155	5106
Engine grid	4161	5 350	4320

K_i is 1 for the processors of cluster c_1 and nearly 3 for the processors of cluster c_2. This means that adding the 16 processors of Vega to the 16 processors of Imbeo can be seen as an increase of about 25% of the computational power of Imbeo. The total execution times obtained for a complete engine cycle on each cluster and also for the distributed simulation, when the workload factors computed by our algorithm are used, are reported in Table 3. For the sake of brevity, we report here only the values obtained when the multi-solver approach is used. Note that numerical results obtained with both the ODE solvers in distributed simulations are comparable with the results of sequential and parallel simulations. We can observe a performance improvement ranging from 13% for the *Case* 2 to 18% for the *Case* 1 with respect to the best parallel performance, that is the performance obtained on 16 processors of Imbeo. This performance improvement can be considered an interesting result, taking into account that our Grid platform is based on a non-dedicated (Internet) interconnection network. Finally, it is worth noting that the time needed by the MJMS system to locate the target resources and to submit the jobs is negligible with respect to the execution time of our simulations.

6. Conclusions

In this work we present results related to the use of an adaptive multisolver for effective solution of stiff ODE systems arising in detailed chemical reaction modeling of diesel combustion. This approach has been implemented for using heterogeneous distributed resources of modern Grid environments, composed of different computers operated in different sites, for distributed simulation of engine applications. Distribution of workload among the computers is based on a strategy that takes into account both model features, such as local stiffness and use of adaptive solvers, and platform features, such as different computational power of processors. Results on small size test cases show that the multi-solver approach realizes a good tradeoff between reliability of simulations and computational efficiency. Furthermore, interesting speedup can be obtained by distributed executions of the MPI-based combustion solver. Future work will be devoted to analyze the performance of our approach on larger problems and larger Grid platforms.

References

[1] A. Ali, G. Cazzoli, S. Kong, R. Reitz, C.J. Montgomery, Improvement in computational efficiency for HCCI engine modeling by using reduced mechanisms and parallel computing, in: Proc. 13th International Multidimensional Engine Modeling User's Group Meeting, 2003.
[2] A.A. Amsden, KIVA-3V: A Block-Structured KIVA Program for Engines with Vertical or Canted Valves, Los Alamos National Laboratory Report No. LA-13313-MS, 1997.
[3] U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, Philadelphia, 1998.
[4] P. Belardini, C. Bertoli, S. Corsaro, P. D'Ambra, Parallel simulation of combustion in common rail diesel engines by advanced numerical solution of detailed chemistry, in: M. Primicerio, R. Spigler, V. Valente (Eds.), Applied and Industrial Mathematics in Italy, World Scientific Pub, Singapore, 2005, pp. 112–123.
[5] P. Belardini, C. Bertoli, S. Corsaro, P. D'Ambra, The impact of different stiff ODE solvers in parallel simulation of diesel combustion, in: Proc. HPCC'05, in: Lecture Notes in Computer Science, vol. 3726, Springer, Berlin, 2005, pp. 958–968.
[6] P. Belardini, C. Bertoli, S. Corsaro, P. D'Ambra, A multi-method ODE software component for parallel simulation of diesel engine combustion, in: Proc. SIAM Conference on Parallel Processing for Scientific Computing, San Francisco, 2006.
[7] P. Belardini, C. Bertoli, V. Fraioli, Chemical markers for multi-method ODE solvers in diesel engines combustion modeling, in: Proc. Internal Combustion Engine Conference, Naples, 2006.
[8] P.N. Brown, G.D. Byrne, A.C. Hindmarsh, VODE: A variable coefficient ODE solver, SIAM J. Sci. Stat. Comput. 10 (1989).
[9] J. Frey, F. Gregoretti, G. Laccetti, A. Murli, G. Oliva, Multi-site jobs management system (MJMS): A tool to manage multi-site MPI applications execution in grid environment, in: Proc. HPDC'15, Workshop on HPC Grid Programming Environments and Components (HPC-GECO/CompFrame), IEEE Computer Society, 2006.
[10] C.W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs NY, 1973.
[11] J. Gustavsson, V.I. Golovitchev, Spray Combustion Simulation Based on Detailed Chemistry Approach for Diesel Fuel Surrogate Model, Society for Automotive Engineers, SAE Paper 2003-0137, 2003.
[12] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems, second edition, Springer Series in Comput. Mathematics, Berlin, 1996.
[13] R.J. Kee, F.M. Rupley, J.A. Miller, CHEMKIN-II: A Fortran Chemical Kinetics Package for the Analysis of Gas-phase Chemical Kinetics, SAND89-8009 Sandia National Laboratories, 1989.
[14] L. Liang, S. Kong, C. Jung, R. Reitz, Development of a semi-implicit solver for detailed chemistry in internal combustion engine simulations, J. Eng. Gas Turbines Power 129 (2007) 271–278.
[15] D. Manca, G. Buzzi-Ferraris, T. Faravelli, E. Ranzi, Numerical problems in the solution of oxidation and combustion models, Combust. Theory Modelling 5 (2001) 185–199.
[16] P.K. Senecal, E. Pomraning, K.J. Richards, T.E. Briggs, C.Y. Choi, R.M. McDavid, M.A. Patterson, Multi-dimensional Modeling of Direct-Injection Diesel Spray Liquid Length and Flame Lift-off Length using CFD and Parallel Detailed Chemistry, Society for Automotive Engineers (SAE) Paper 2003-01-1043, 2003.