Cambridge Centre for Computational Chemical Engineering

University of Cambridge

Department of Chemical Engineering

Preprint ISSN 1473 – 4273

A new explicit numerical scheme for large scale combustion problems

Sebastian Mosbach[1], Markus Kraft[1]

submitted: July 15, 2003

[1] Department of Chemical Engineering
University of Cambridge
Pembroke Street
Cambridge CB2 3RA
UK
e-Mail: sm453@cam.ac.uk,
markus_kraft@cheng.cam.ac.uk

Preprint No. 12

c4e

Key words and phrases. systems of stiff ordinary differential equations, explicit method, combustion.

Edited by
Cambridge Centre for Computational Chemical Engineering
Department of Chemical Engineering
University of Cambridge
Cambridge CB2 3RA
United Kingdom.

Fax: + 44 (0)1223 334796
E-Mail: c4e@cheng.cam.ac.uk
World Wide Web: http://www.cheng.cam.ac.uk/c4e/

Abstract

This paper introduces a new explicit numerical method with adaptive time stepping which is suited to large and stiff systems of ordinary differential equations (ODEs). The algorithm, which is motivated by the theory of Markov jump processes, is very simple and can be easily programmed. Various numerical experiments are performed to assess the efficiency of the algorithm. For this the ignition of a stoichiometric mixture of n-decane and air at constant pressure and temperature is modelled using a large system of ordinary differential equations containing 1218 strongly coupled and stiff equations. The new algorithm is compared to the software packages DASSL and LSODE, which are shown to be outperformed for moderate precision. The numerical experiments also indicate that the approximate solution obtained from the new algorithm converges to the exact solution of the ODE.

Contents

1 Introduction

The numerical solution of large systems of ordinary differential equations (ODEs) has attracted considerable interest in the last few decades. Good reviews can be found in references [2] and [4]. This can be attributed to the fact that many models for industrial and physical processes are formulated in this way. One example studied in this paper is the modelling of fuels consisting of higher hydrocarbons. The detailed chemical mechanisms derived for these fuels often contain up to a thousand chemical species and several thousand chemical reactions occurring on time scales that can differ by many orders of magnitudes. Hence, the chemical source terms in detailed combustion models are very stiff [4, 2, 8].

For such models numerical methods based on implicit schemes have been developed. Implicit methods have very good stability properties which combined with adaptive time stepping lead to very efficient numerical algorithms. However a nonlinear system of equations has to be solved for every time step which involves manipulating the Jacobi matrix. This necessitates the use of linear algebra routines, which can become very expensive for large systems. Explicit methods do not require the costly solution of a nonlinear system of equations but suffer from severe stability problems which lead to extremely short time steps. Hitherto explicit methods have been developed only for mildly stiff systems (see [12] and references therein). Very stiff problems could not be solved at reasonable computational cost.

The **purpose of this paper** is to present a new explicit scheme for solving a large system of stiff ordinary differential equations as they appear in the combustion of fuels containing higher hydrocarbons. This new explicit scheme is very efficient and very easy to implement. Despite its explicit nature it is tailored to large and stiff systems of ODEs but can also be used for solving (at moderate accuracy) any initial value problem (IVP) of the form

$$\frac{d}{dt}x_j(t) = Q_j\big(x(t)\big) \qquad ; j \in \{1, \ldots, S\} \tag{1}$$

with the initial condition

$$x(0) = x^0 \in \mathbb{R}^S.$$

For testing purposes we consider the combustion of a stoichiometric mixture of n-decane and air at constant pressure and temperature in a plug flow reactor. The efficiency of the algorithm is studied by comparing the CPU time and errors with the state-of-the-art software packages DASSL [1] and LSODE [9].

2 The algorithm

The new algorithm is motivated by the theory of Markov jump processes. It can be interpreted as a deterministic version of such a process. More details on the general relationship between Markov jump processes and the numerical treatment of ODEs

can be found in reference [6]. The algorithm considered here can be regarded as a simple representative of a whole class of numerical algorithms, which is closely related to the Euler scheme but possesses the following important differences:

1. In each step, not all components are altered, but only those whose predicted change exceeds a certain value.
2. The changes in the chosen components are constant and are directly connected to the time step. In other words, the solution is approximated by a step function with jumps of a predefined size, where the time interval between the jumps is intimately related to their magnitude.
3. The time stepping is adaptive and is motivated by the mean waiting time of the underlying jump process.

In the same way as the Euler algorithm is the starting point for many explicit schemes that led, for example, to Runge-Kutta type schemes, the new algorithm can be improved in several ways. However, in this paper we chose to study this generic algorithm to demonstrate how successful even this simple explicit scheme can be, when applied to problems that, so far, could only be solved with very sophisticated software packages.

Our algorithm reads in algorithmic language:

1. Fix $\texttt{ATOL} > 0$, a stopping time $t_{\text{stop}} \geqslant 0$, initialise the solution vector $x = x^0$ and the change vector $\Delta x = 0 \in \mathbb{R}^S$.
2. Update the change vector
$$\Delta x \mapsto \Delta x + \Delta t \times Q = \Delta x + \frac{\texttt{ATOL}}{\sum |Q_i|} \times Q,$$
where Q is the right hand side of equation (1).
3. For each $j \in \{1, \ldots, S\}$, if $|\Delta x_j| \geqslant \texttt{ATOL}$ then update
$$\begin{aligned} x_j &\mapsto x_j + \operatorname{sign}(\Delta x_j) \times \texttt{ATOL} \\ \Delta x_j &\mapsto \Delta x_j - \operatorname{sign}(\Delta x_j) \times \texttt{ATOL}. \end{aligned}$$
4. Update $t \mapsto t + \Delta t$ with $\Delta t = \texttt{ATOL} / \sum |Q_i|$.
5. If $t < t_{\text{stop}}$ then go to step 2.

In this algorithm ATOL is the user defined *a priori* absolute error tolerance. The change vector Δx is updated in the second step and corresponds directly to the Euler method. In step three the method starts to differ from the Euler approach. Each component is checked whether it leaves a prescribed interval given by the error tolerance. This concept implies that variations in the solution vector smaller

than ATOL cannot be accounted for. In the current version of the algorithm the same error tolerance is chosen for all components but if the components differ in magnitude one can introduce an error tolerance for each component separately. For the sake of simplicity we will not follow this route. All components that have been updated require an appropriate adjustment of the change vector. This step ensures the correct selection of components, which is motivated by the selection probability of each component in the corresponding stochastic process [6]. In step four the time is updated according to $\Delta t = \texttt{ATOL}/\sum |Q_i|$, which is derived from the mean waiting time of the corresponding Markov jump process. Consequently, the time step Δt is adjusted to comparatively small/large values whenever the magnitude of the right hand side is large/small (i.e. the solution is changing rapidly/slowly).

It is clear that for "bookkeeping" purposes the (constant) factor of ATOL can be dropped from the update of the changes Δx. A FORTRAN 95 source code implementing this algorithm could look like the following:

```
Deltax = 0d0                    !clear change vector
DO WHILE (t<tstop)              !time loop
  CALL rhs(x,Q)                 !calculate right hand side
  QSum = SUM(ABS(Q))            !calculate sum
  Deltax = Deltax+Q/QSum        !update change vector
  !update quantities and changes if necessary
  DO j = 1,S
    IF (ABS(Deltax(j))>=1d0) THEN
      x(j) = x(j)+SIGN(ATOL,Deltax(j))
      Deltax(j) = Deltax(j)-SIGN(1d0,Deltax(j))
    ENDIF
  ENDDO
  t = t+ATOL/QSum               !update time
ENDDO
```

The subroutine rhs(x,Q) evaluates the right hand side Q of equation (1) at x. Note that in the case $\texttt{QSum} = \sum |Q_i| = 0$ the algorithm has to terminate since the solution is constant, which has to be intercepted appropriately. This algorithm is not significantly longer than Euler's method and very easy to implement. To the best knowledge of the authors, such a deterministic algorithm using constant jumps together with automatic time adaptivity has not been considered previously.

3 Numerical study

In this section we study the numerical properties of our algorithm and compare it to several other numerical methods. The computations were carried out on an Intel Pentium III PC at 866MHz running Microsoft Windows 2000. The executable was created using a COMPAQ FORTRAN compiler.

Table 1: *Considered reaction mechanism*

Name	#Species	#Reactions	Reference	Website
n-decane	1218	4825	[3]	www.ensic.u-nancy.fr/DCPR/Anglais/GCR/softwares.htm

Table 2: *Initial conditions for the test system*

Mechanism	Species	Mole fraction	Temp. [K]	Pressure [Pa]
n-decane	$nC_{10}H_{22}$	0.01346	1500	1.01325×10^5
	O_2	0.20867		
	N_2	0.77787		

In order to choose an appropriate chemical system we have examined seven different combustion systems with an increasing number of species and reactions. From these preliminary studies we chose the stoichiometric combustion of n-decane. With the increasing effort of scientists to model fuels consisting of higher hydrocarbons, detailed chemical mechanisms of this size will become more common in future. The chemical model is specified in **Table** 1. The initial conditions given in **Table** 2 complete the initial value problem. It comprises 1218 equations which are strongly coupled. The elementary reactions introduce a spectrum of time scales which vary over many orders of magnitude. Hence the system of ODEs is very stiff and an ideal test case for our method. The simulation time $t_{\text{stop}} = 10^{-3}$ s is chosen to ensure ignition has been captured and the chemical system is sufficiently close equilibrium.

In order to obtain an accurate numerical solution for our IVP, and to assess the efficiency of the new method, various other numerical methods were tested. In all cases we made use of the CHEMKIN libraries [5] to evaluate the chemical source terms. When attempting to solve the IVP with standard explicit methods like Euler or fourth order Runge-Kutta one encounters the following problems. Using constant time steps, both methods are unable to produce any non-divergent approximation unless the step size is chosen extremely small. For instance, at a step size of 10^{-12} Euler's method is estimated to take almost one year to calculate up to $t_{\text{stop}} = 10^{-3}$ s. Furthermore, conventional step size adaption techniques are found to reduce the step size quickly to exceedingly small values (10^{-15} or smaller) even for mild accuracy requirements. The method by Runge and Kutta exhibits only marginally better performance for both adaptive and non-adaptive modes of operation. This kind of behaviour is well-known for stiff systems and is pointed out in many texts on numerical ODE solution (e.g. [4]). We also employed the code ROCK4 (see **Table** 3) which is an explicit fourth order Runge-Kutta method with extended stability domain along the negative real axis designed for mildly stiff differential equations. We were not able to obtain a solution to our problem with this code. Finally, we de-

cided to use the software packages DASSL and LSODE. Both codes are frequently used and widely accepted [2, 10] and can therefore be considered as state-of-the-art numerical tools for combustion problems similar to the IVP described above. The references as well as links to websites where these packages can be obtained are given in Table 3. For large combustion systems such as the one considered in this paper, Schwer *et al.* [10] enhanced the performance of DASSL using the software package DAEPACK. This package includes automatic differentiation and determines sparsity patterns. Due to the third body reactions in the chemical mechanism it would have been necessary to reformulate the IVP to achieve a good degree of sparsity. Only then could a significant speed up have been obtained. Since their method only applies to chemical systems whereas ours is intended for general use, we decided not to take this route. We also decided not to follow the hybrid approach suggested by Valorani and Goussis [11]. They use the concepts embodied in the computational singular perturbation (CSP) method. However they tested this method on a relatively small combustion problem, the ignition of a methane/air mixture (49 species and 260 reactions). They found that their method was significantly slower than LSODE. Since their method makes use of the eigenvalues of the Jacobian its efficiency is expected to decrease for systems of the size we are interested in this paper.

Table 3: *Considered ODE solver packages*

Name	Reference	Website
DASSL	[1]	www.engineering.ucsb.edu/%7Ecse/
LSODE	[9]	www.llnl.gov/CASC/download/download_home.html
ROCK4	[4]	www.unige.ch/math/folks/hairer/software.html

Table 4: *Conversion table for the values of* `ATOL`

Index	DASSL	Index	LSODE	Index	new algorithm
a	1.00×10^{-11}	A	1.00×10^{-11}	1	5.00×10^{-11}
b	5.00×10^{-11}	B	5.00×10^{-11}	2	1.00×10^{-10}
c	2.00×10^{-10}	C	2.00×10^{-10}	3	2.00×10^{-10}
d	1.00×10^{-10}	D	1.00×10^{-10}	4	5.00×10^{-10}
e	5.00×10^{-10}	E	5.00×10^{-10}	5	1.00×10^{-9}
f	1.00×10^{-8}	F	1.00×10^{-9}	6	2.00×10^{-9}
g	1.00×10^{-9}	G	2.00×10^{-9}	7	2.50×10^{-9}
h	1.00×10^{-6}	H	5.00×10^{-9}	8	3.33×10^{-9}

The new algorithm possesses one free input parameter, namely the absolute error tolerance `ATOL`. In order to compare the performance of the new algorithm the rela-

tive tolerances RTOL in DASSL and LSODE are chosen to be zero for the numerical experiments. We found that for the test system considered, both packages perform best for RTOL $= 0$. Eight numerical experiments for each method with different values of ATOL are presented here. **Table** 4 displays an index and the corresponding error tolerance for each solver. Besides these experiments a high precision run was performed using DASSL with RTOL $= 10^{-9}$ and ATOL $= 10^{-15}$. In the following, the result of this run is referred to as the "exact" solution. As an estimate of the global error we measure the absolute overall deviation of the approximation $\tilde{x}$ from the exact solution x of the ODE after the calculation has been performed (i.e. *a posteriori*). Specifically, we employ

$$c_{\mathrm{tot}} := \frac{1}{M+1} \sum_{j=0}^{M} \left| x(t_j) - \tilde{x}(t_j) \right|, \tag{2}$$

where the time interval $[0, t_{\mathrm{stop}}]$ is split into M subintervals of equal length via

$$t_j := j \times \frac{t_{\mathrm{stop}}}{M}.$$

For the numerical study $M = 2^9 = 512$ has been used. Formula (2) can also be used for any quantity f which is a function of the solution by substituting f for x. In particular, since the mass density ϱ is a function of all chemical species, it was chosen for evaluating the global *a posteriori* error c_{tot}. For each entry in Table 4, one run was performed. For each of these runs the CPU time was recorded and the error in the density was calculated. The results are displayed in **Figure** 1, where each point is labelled with an index whose corresponding value of ATOL can be obtained from Table 4. From Figure 1 it is clear that the numerical effort does not change significantly for different degrees of precision in the case of LSODE and DASSL. Both packages perform similarly well for this test case. In contrast to the other explicit methods considered we were able to obtain a numerical solution with the new algorithm. For moderate precision the new algorithm is found to be faster than DASSL or LSODE. **Figure** 2(a) illustrates how the density calculated for various values of ATOL given in Table 4 converges to the exact solution. For any smaller values of ATOL the result from the new algorithm coincides with the exact density and cannot be distinguished by visual inspection. **Figure** 2(b) reveals that the dependence of the number of steps on ATOL is given (for sufficiently small values of ATOL) by the power law $a\mathtt{ATOL}^b$ (with $a = (2.7 \pm 0.9) \times 10^{-4}$ and $b = -0.870 \pm 0.015$ in this case). Due to the structure of the algorithm, the CPU-time is roughly proportional to the number of steps, which can also be established by combining Figures 1 and 2(b). For the current IVP we used the following strategy for selecting a reasonable value of ATOL. Starting with a large value of ATOL, say 10^{-5}, we then decreased ATOL in successive steps by an order of magnitude each. Monitoring the total number of occurred events (or equivalently the CPU-time) the algorithm behaves at some point according to the power law noticed above, which can be identified most easily in a log-log plot like Figure 2(b). Once this convergence is established, the numerical experiments indicate that the produced approximation is "sufficiently close" to

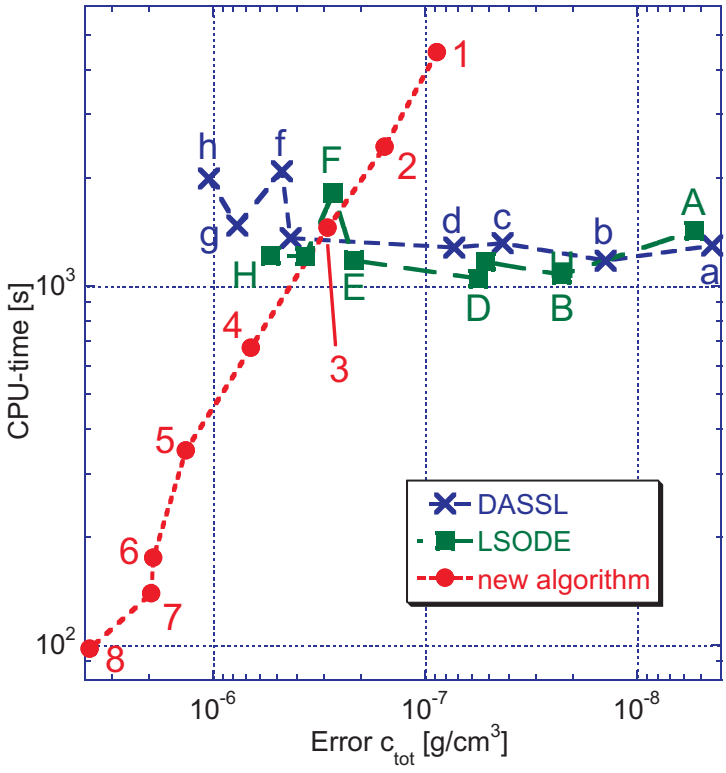
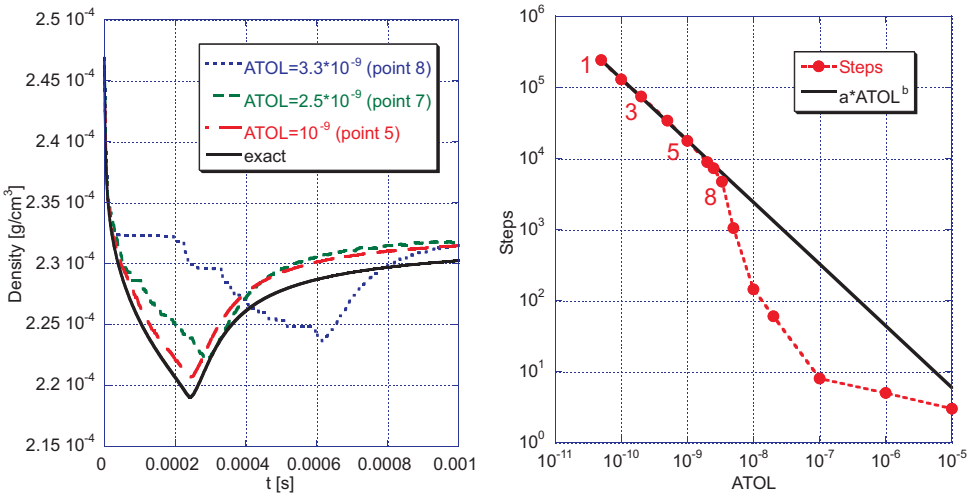


Figure 1: *CPU-time of the considered algorithms as a function of the total error c_{tot} of the mass density ϱ for the n-decane mechanism*

the exact solution. In **Figure** 3(a) the time evolution of some major species is displayed. This result was obtained with the new algorithm choosing $\texttt{ATOL} = 2 \times 10^{-10}$ (corresponding to point 3 in Figure 1). The profiles from this simulation coincide with the exact solution. In **Figure** 3(b), methane and the OH radical are shown, which have been chosen as examples of species whose concentration exhibits sharp gradients and small absolute values. Even for these species very good agreement between the exact solution and the approximation by new algorithm can be found, for this particular choice of `ATOL`. This also suggests that the error in the density is a good indicator for the error in the species concentrations. **Figure** 4 shows how the automatic step size adaption works in practice (for point 3). Since the total number of steps for this run is 74476 we decided to plot the step size not for every single step. Instead, the solution interval $[0, t_{\text{stop}}]$ is (as above) split into $M = 512$ subintervals over each of which the maximum, average and minimum step size ($h_{\max}$, h_{avg} and $h_{\min}$ respectively) is calculated. For the maximum and minimum curves only every twentieth data point is shown. In this diagram, one can recognise the following characteristic features, which are desirable for any kind of step size adaption. At $t = 0$ and during the ignition (at $t \approx 2.5 \times 10^{-4}$ s) the step size becomes relatively small, because at these points in time the concentrations of several species are rapidly changing (the right hand side Q of equation (1) is large in magnitude). Recall that the step size is inversely proportional to the sum of the absolute values of the components of the right hand side. At later times the step size becomes com-



(a) Time evolution of the mass density for various values of `ATOL`

(b) Number of steps as a function of `ATOL`

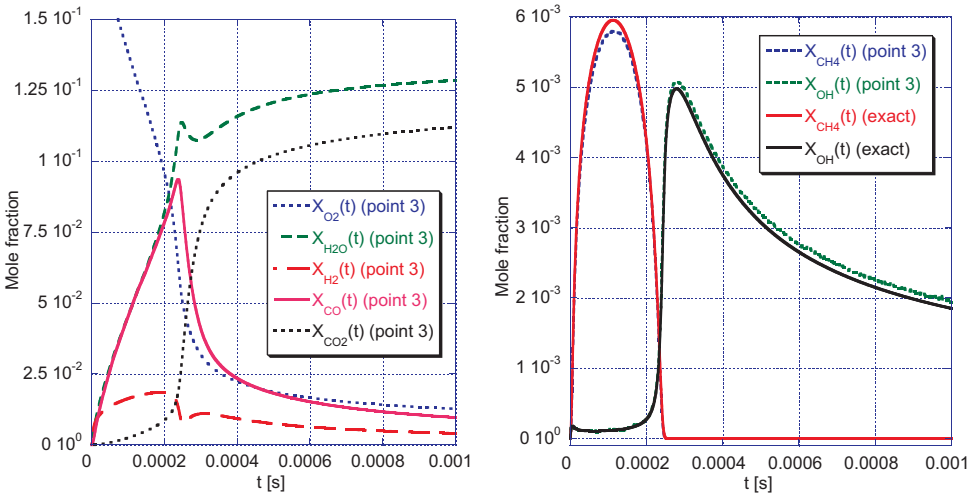
Figure 2: *Convergence properties of the new algorithm*

paratively large as the system approaches equilibrium (i.e., small right hand side). Note also the large fluctuations, which reflect the sensitivity of the right hand side towards the quantities of interest.

Even though these numerical tests are not exhaustive, one is tempted to conclude that it is the idea of treating each component of the solution separately which explains the relatively high performance and suitability for stiff systems. From Figure 1 it is also clear that this algorithm is only suitable for situations where moderate precision of the solution is sought. However, there are many situations where high precision is not required, in particular when a system of ODEs must be solved in the context of operator splitting (see for example [7]), where it is important that the start up costs of the ODE solver are low. This is the case for the new algorithm due to its simple explicit structure. However, a great advantage of the new algorithm is that the numerical effort scales linearly with the number of equations. This means that our algorithm is an ideal candidate for very large and very stiff systems of ordinary differential equations. Such systems occur frequently in dynamic process simulation of large scale plants in the chemical industry and electronic circuit design. Other examples can be found in the numerical treatment of some partial differential equations.

4 Conclusion

We have introduced and tested a new explicit algorithm for large (and possibly stiff) systems of ordinary differential equations. This algorithm is simple and can be easily implemented on a computer. It has no start-up cost, has automatic time stepping,



(a) Major species

(b) Methane and the OH radical compared to the "exact" solution

Figure 3: *Time evolution of selected species of the n-decane mechanism*

and scales linearly with the number of equations.

We assessed the performance of the new algorithm by studying the isothermal ignition of a stoichiometric mixture of n-decane and air at constant pressure. The corresponding model contains more than 1000 strongly coupled and very stiff differential equations. We considered several numerical methods as possible candidates for assessing the new algorithm, and chose the software packages DASSL and LSODE to gauge its performance.

Several numerical experiments were performed. We found that the new algorithm converges and the number of steps has a power law dependence on the *a priori* error tolerance `ATOL`. The global *a posteriori* error was measured by the difference in the density between a high precision run, obtained from DASSL, and the new method. We established that the CPU-time also obeys a power law with respect to `ATOL` due to the proportionality of the number of steps to the CPU-time. For moderate precision the new algorithm outperforms both DASSL and LSODE. In addition, the automatic step adaption was shown to be reasonable. Small steps are taken during ignition and larger steps towards chemical equilibrium.

Although the numerical experiments demonstrate that the new algorithm works in practice, there are open questions beyond the scope of this paper. We have not established a theoretical convergence result. Another issue that remains to be clarified concerns the fact that this new explicit algorithm performs so well. Theoretical stability results are required which provide an explanation of the robustness of the new algorithm found in the numerical experiments that were carried out in this paper.

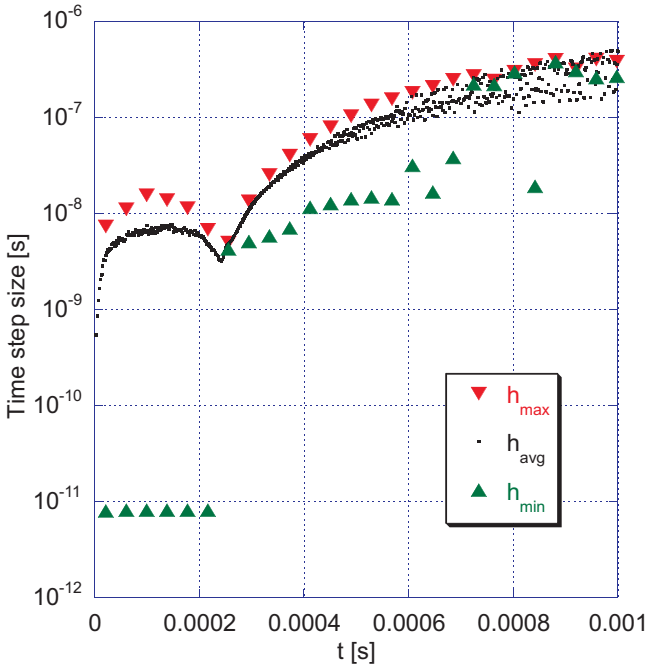


Figure 4: *Automatic adaption of the time step size (point 3)*

Acknowledgements

This work was funded by the EPSRC (grant number GR/R85662/01) under the title "Mathematical and Numerical Analysis of Coagulation-Diffusion Processes in Chemical Engineering".

References

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold. Numerical solution of initial-value problems in differential-algebraic equations. *SIAM Classics in Applied Mathematics*, 14, 1996.

[2] J. R. Cash. Efficient numerical methods for the solution of stiff initial-value problems and differential algebraic equations. *Proc. R. Soc. Lond.*, 459:797–815, 2003.

[3] P. A. Glaude, V. Warth, R. Fournet, F. Battin-Leclerc, G. Scacchi, and G. M. Côme. Modelling of the oxidation of n-octane and n-decane using an automatic generation of mechanisms. *Int. J. Chem. Kin.*, 30:949–959, 1998.

[4] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer Verlag, Berlin Heidelberg New York, second revised edition, 1996.

[5] R. J. Kee, F. M. Rupley, E. Meeks, and J. A. Miller. Chemkin-III: A FORTRAN chemical kinetics package for the analysis of gas-phase chemical and plasma kinetics. Technical Report SAND96-8216 UC-405, Sandia National Laboratories, 1996.

[6] S. Mosbach and M. Kraft. Stochastic modelling of large scale combustion systems. Technical Report 13, Cambridge Centre for Computational Chemical Engineering, 2003.

[7] R. D. Mott, E. S. Oran, and B. van Leer. A quasi-steady-state solver for the stiff ordinary differential equations of reaction kinetics. *J. Comp. Phys.*, 164:407–428, 2000.

[8] E. S. Oran and J. P. Boris. *Numerical Simulation of Reactive Flow*. Cambridge University Press, 2nd edition, 2000.

[9] K. Radhakrishnan and A. C. Hindmarsh. Description and use of LSODE, the Livermore Solver for Ordinary Differential Equations. Technical Report UCRL-ID-113855, Lawrence Livermore National Laboratory, 1993.

[10] D. A. Schwer, J. E. Tolsma, W. H. Green, and P. I. Barton. On upgrading the numerics in combustion chemistry codes. *Combustion and Flame*, 128:270–291, 2002.

[11] M. Valorani and D. A. Goussis. Explicit time-scale splitting algorithm for stiff problems: Auto-ignition of gaseous mixtures behind a steady shock. *J. Comput. Phys.*, 169(1):44–79, 2001.

[12] J. G. Verwer. Explicit Runge-Kutta methods for parabolic partial differential equations. *Appl. Numer. Math.*, 22(1-3):359–379, 1996.