Cambridge Centre for Computational Chemical Engineering

University of Cambridge

Department of Chemical Engineering

Preprint ISSN 1473 – 4273

Stochastic solution of ordinary differential equations

Sebastian Mosbach[1], Markus Kraft[1]

submitted: December 4, 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. 13

c4e

Key words and phrases. ODE, Markov process, 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

In this paper new stochastic algorithms for the numerical solution of systems of ordinary differential equations (ODEs) are proposed. Furthermore, a correspondence principle is established between these algorithms, which are based on the theory of Markov jump processes, and deterministic schemes.

For one of the proposed stochastic algorithms, a detailed numerical study of some of its properties is carried out using homogeneous gas-phase reaction mechanisms describing the combustion of hydrogen, carbon monoxide, methane, n-heptane, iso-octane and n-decane. Fuels like mixtures of n-heptane and iso-octane can contain up to a thousand species and several thousand reactions and are of practical relevance to industrial combustion processes.

One deterministic method yielded by our correspondence principle has been proposed and studied in a previous paper and is used here in order to shed light on various aspects of the considered stochastic algorithm. In addition, we use the widespread state-of-the-art stiff ODE-solver package DASSL for comparison.

Advantages of our methods include among others their exceptional simplicity of implementation, negligible start-up costs and, as shown by numerical experiments, a linear scaling behaviour of the computational time with the number of equations.

It is also shown that for large systems, the proposed algorithms exhibit computational efficiency similar to conventional implicit solvers, assuming multiple runs in the stochastic case. In view of the stiffness of the considered systems and the explicit nature of our algorithms, this is rather surprising.

These properties suggest as typical application large operator-splitting problems requiring moderate accuracy, such as PDF transport models.

Contents

1 Introduction

Turbulent combustion is ubiquitous in today's technology, for example in gas turbines, diesel and spark-ignition engines. An important issue in combustion technology in general is its detrimental impact on the environment through unwanted emissions of combustion products and by-products. Industry employs numerical modelling of these processes to achieve a deeper understanding in order to reduce combustion emissions. One modelling approach that has been shown to be successful is based on the probability density function (PDF) transport equation of the physical quantities of interest [28]. In the special case of spacial homogeneity these PDF methods reduce to stochastic reactor models (SRMs), which can be found in many industrial applications (see for example [1, 19, 23]).

Some of the earliest works on stochastic methods applied to chemical kinetics include [22, 5, 12] and [31]. In these papers, stochastic direct simulation algorithms are presented, which predict the time evolution of homogeneous chemically reacting gas mixtures. As pointed out in [13], such algorithms of stochastic nature are able to account for inherent microscopic fluctuations. In [14], a derivation of the chemical master equation is given showing that this equation describes exactly any well-stirred and thermally equilibrated gas-phase chemical system. More recently, in [20] a direct method to calculate the combustion of practically relevant fuels is used. Improvements of the direct simulation strategy in physical and chemical problems are indicated for instance in [4] and [15].

The application of the theory of Markov processes to the solution of ODEs was initiated by [21], though from a purely theoretical point of view without numerically exploiting any of the ideas. We are not aware of any attempts to investigate the numerical solution of ODEs by stochastic methods.

A good recent review of available solvers for stiff initial value problems is given in [6]. There, the solver DASSL, which is also used in this paper, is referred to as a very powerful method and is recommended for general use when approaching stiff ODEs or differential/algebraic equations.

In [30] it is noted that combustion mechanisms, which are expected to become increasingly large in the near future, are rather sparse, due to the fact that each species reacts at most with a small fraction of the total number of species. Also, sparse routines for DASSL are developed.

The purpose of this paper is to present numerous algorithms for the solution of systems of ODEs by means of sequences of Markov processes. In addition, we propose a scheme that establishes a correspondence between conventional and stochastic algorithms yielding numerous examples of deterministic as well as stochastic methods, all of which are explicit and therefore remarkably simple to implement. However, it is beyond the scope of this text to explore all possible aspects. Our main intention is to present the algorithms, their underlying principles and to demonstrate for a particular one its performance in practically relevant applications. This is put into practice by studying one of the proposed stochastic algorithms and a version of its deterministic analogue numerically in the context of some homogeneous gas-phase reaction mechanisms describing the combustion of small hydrocarbons. We also

show how our algorithms compare to DASSL, a state-of-the-art solver package for stiff systems.

This paper is structured as follows. In section 2 we give some theoretical background of stochastic ODE solution. Subsection 2.1 specifies which class of ODEs we intend to solve by a stochastic approach and subsection 2.2 elucidates how an ODE can be approximated by a sequence of Markov processes. A formal derivation of how to specify Markov processes which describe ODEs is given in subsection 2.3. Based on the understanding of a Markov jump process, one can easily write down algorithms that produce approximate solutions, which is done in a number of ways in section 3. In subsection 3.1, the general scheme is explained with a number of example applications in subsection 3.2. We then elaborate on connections between stochastic and deterministic approaches in subsections 3.3 and 3.4. In the course of the paper, we come across a variety of different algorithms, whose performance (i.e. the ratio between accuracy and computational expense) remains to be investigated. Due to time and space limitations, we had to restrict our numerical experiments, whose results are presented in section 4, to two algorithms. First, we specify the considered test systems of ODEs in subsection 4.1, for which we chose the combustion systems hydrogen, carbon monoxide, methane, n-heptane, iso-octane and n-decane. In subsection 4.2, we then write down the precise algorithms studied. In subsection 4.3 we give various definitions concerning numerical errors that are used for assessing the quality of approximations. We then discuss the performed experiments and their results in section 4.4. Finally, in section 5 we draw some conclusions.

2 Theoretical background

2.1 The initial value problem

Consider the system of $S \in \mathbb{N}$ ordinary differential equations

$$\frac{d}{dt}\lambda_j(t) = Q_j\big(\lambda(t)\big) \qquad ; j \in \{1, \ldots, S\} \tag{2.1}$$

with the initial condition

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

The numerical algorithms presented in this work can be applied to problems with Lipschitz continuous right hand side $Q : \mathbb{R}^S \to \mathbb{R}^S$, which is a weaker requirement compared to the differentiability needed by most conventional solvers. We furthermore assume that the initial value problem possesses a unique solution on some compact interval. Also, we consider only problems without explicit time dependence.

2.2 Approximation by Markov processes

Given a physical system whose states can be described by S quantities $x_1, \ldots, x_S$. In other words, the states $x = (x_1, \ldots, x_j, \ldots, x_S)$ of the system are represented by

vectors in the state space $\mathbb{R}^S$. Assume furthermore that there exist I different *jumps* $J_1, \ldots, J_\alpha, \ldots, J_I$ describing transitions from a current state x to a new state $J_\alpha(x)$, where the α-th jump occurs at the *rate* (number of jumps per unit time) $R_\alpha(x) \geqslant 0$.

$$J_\alpha : \mathbb{R}^S \ni x \mapsto J_\alpha(x) = x + y_\alpha(x) \in \mathbb{R}^S \tag{2.2}$$

Greek indices refer to the jumps whereas Latin indices refer to the quantities themselves. The transitions between the states occur at random with probability distribution determined by the rates. Mathematically, such a system is modelled by a Markov jump process Z with *generator*

$$(\mathcal{A}\Phi)(x) = \sum_{\alpha=1}^{I} R_\alpha(x)\big[\Phi\big(J_\alpha(x)\big) - \Phi(x)\big] \qquad ; x \in \mathbb{R}^S, \tag{2.3}$$

which contains all of the above information. In this formula, Φ denotes a test function from $\mathbb{R}^S$ to $\mathbb{R}$. The index α in the sum runs over all possible jumps which the process can perform, starting from the current state x.

The question which rates and jumps have to be used in order to specify a Markov jump process that approximates an ODE is answered in subsection 2.3. In section 3 it is shown how to translate this description into an algorithm.

Usually, the system given by (2.3) is just an approximation of a problem of interest, which requires a means for controlling the accuracy of the approximate solutions. In many applications, the jumps J_α and the rates R_α depend on some error control parameter, for instance the local absolute error tolerance `ATOL` in the ODE case, or, if the system contains a particle ensemble, the particle number n. In probability theory, this is summarized under the concept of a sequence $(Z^{(n)})_{n\in\mathbb{N}}$ of Markov jump processes, where the index n can be regarded as the approximation parameter. In the limit $n \to \infty$ a sample path of the Markov process "is" the exact solution of the considered problem. This notion of convergence of a sequence of Markov processes to the solution of an ODE-system is made precise by the so-called *fluid limit theorem*, originally published by Kurtz [21], then extended and generalized by Darling and Norris [10]. For introductory approaches, the reader is referred to [9] or [27], and further material can be found for example in [11].

2.3 The master equation

The master equation describes the time evolution of the probability density function $f(x,t)$ of a Markov process. It is assumed to be given by

$$\frac{\partial}{\partial t} f(x,t) = \mathcal{A}^* f(x,t), \tag{2.4}$$

where the formal adjoint $\mathcal{A}^*$ of the generator is defined via

$$\int \varphi(x)(\mathcal{A}\Phi)(x)dx = \int (\mathcal{A}^*\varphi)(x)\Phi(x)dx. \tag{2.5}$$

Starting from equation (2.4), we motivate a number of generators, first in their one-dimensional form. Then, these considerations are generalized to an arbitrary number of ODEs in order to be utilized later on to construct stochastic algorithms and their deterministic counterparts.

2.3.1 One-dimensional form

Exact case

Consider the generator

$$(\mathcal{A}\Phi)(x) = Q(x)\frac{d}{dx}\Phi(x) \tag{2.6}$$

of a Markov process Z corresponding to the master equation

$$\frac{\partial}{\partial t}f(x,t) = -\frac{\partial}{\partial x}\Big(Q(x)f(x,t)\Big). \tag{2.7}$$

with initial probability density function $f(x,0) = \delta(x-\lambda^0)$. This initial condition represents the fact that we always start with a single, fixed initial value λ^0. One can show by integrating with respect to x against a test function that a weak solution of equation (2.7) is given by

$$f(x,t) = \delta\big(x-\lambda(t)\big). \tag{2.8}$$

Therefore, the expectation of the process at time t equals the value of the solution λ at time t:

$$\mathsf{E}Z(t) = \int xf(x,t)dx = \lambda(t).$$

The variance of the Dirac delta distribution is zero of course, which means that the sample path (there exists only one, no randomness) of the process generated by (2.6) is the exact solution.

First order finite difference approximation

We now approximate (2.6) by the first order finite difference quotient

$$(\mathcal{A}\Phi)(x) = Q(x)\frac{\Phi(x+h)-\Phi(x)}{h}, \tag{2.9}$$

which looks very similar to the general generator (2.3). However, interpreting (2.9) as a jump process only makes sense if the rates are non-negative, which implies that h has to be positive/negative whenever Q is. This will become relevant when algorithms are derived in section 3.

Second order finite difference approximation

The derivative of a suitably differentiable function can be approximated up to second order by

$$\Phi'(x) = \frac{-\Phi(x+h)+4\Phi(x+h/2)-3\Phi(x)}{h} + O(h^2).$$

Combining this with (2.6) yields

$$(\mathcal{A}\Phi)(x) = Q(x)\frac{-\Phi(x+h)+4\Phi(x+h/2)-3\Phi(x)}{h}. \tag{2.10}$$

Higher order approximations are motivated by the following idea: Assume that the master equation corresponding to a k-th order generator possesses as solution a

probability density function which equals that of the exact equation up to a k-th order correction in h. Then the moments

$$m_j(t) = \int x^j f(x,t) dx$$

of the exact distribution equal those of the approximate distribution also up to k-th order in h. In particular, this implies that the variance

$$\operatorname{Var} Z(t) = m_2(t) - \big[m_1(t)\big]^2$$

of the approximate process Z differs from that of the exact process by terms of k-th order in h. More specifically, since in our case the variance of the exact process vanishes, the variance of the approximate process is proportional to h^k.

2.3.2 General form in S dimensions

Exact case

The S-dimensional version of (2.6) reads

$$(\mathcal{A}\Phi)(x) = \sum_{j=1}^{S} Q_j(x) \frac{\partial}{\partial x_j} \Phi(x). \tag{2.11}$$

As in the one-dimensional case, this has zero variance for an initial delta distribution, i.e. the sample path is the exact solution of the ODE.

First order finite difference approximation

There are numerous ways to extend generator (2.9) to S dimensions, one of the simplest being given by

$$(\mathcal{A}\Phi)(x) = \sum_{j=1}^{S} Q_j(x) \frac{\Phi(x + he_j) - \Phi(x)}{h}.$$

Here, e_j denotes the j-th unit vector in $\mathbb{R}^S$, i.e. $(e_j)_k = \delta_{jk}$, with

$$\delta_{jk} = \begin{cases} 1 & \text{for } j = k \\ 0 & \text{otherwise} \end{cases}$$

the Kronecker delta. This expression can be generalized such that each component possesses "its own h", which may even depend on the current state x, i.e.

$$(\mathcal{A}\Phi)(x) = \sum_{j=1}^{S} \frac{Q_j(x)}{h_j(x)} \big[\Phi\big(x + h_j(x)e_j\big) - \Phi(x)\big]. \tag{2.12}$$

As noticed above, the positivity of the rates is taken into account via an appropriate choice of h_j (see subsection 3.2.1).

3 Algorithms

In this section, a variety of algorithms is presented including a (non-rigorous) correspondence principle between stochastic and deterministic methods.

3.1 General method

Equipped with the knowledge of the rates R_α and the jumps J_α describing a system, in other words knowing generator (2.3), one readily deduces a simulation procedure that creates sample paths of the process, i.e. approximations to the solution of the considered problem.

1. Initialise the state vector x, i.e. set it equal to the given initial values, set $t = 0$ and fix a stopping time $t_{\text{stop}} > 0$.

2. Wait an exponentially distributed time τ with waiting time parameter

$$\pi = \sum_{\beta=1}^{I} R_\beta. \tag{3.1}$$

   That means, advance the time $t \mapsto t + \tau$ such that the waiting time τ is distributed according to[1]

$$\mathsf{P}(\tau \geqslant s) = \exp(-\pi s) \qquad \forall s \geqslant 0. \tag{3.2}$$

3. If $t > t_{\text{stop}}$ then stop.

4. Choose a jump index $\alpha \in \{1, \ldots, I\}$ according to the probability

$$P_\alpha = \frac{R_\alpha}{\sum_{\beta=1}^{I} R_\beta} = R_\alpha/\pi. \tag{3.3}$$

5. Perform the jump according to

$$x \mapsto J_\alpha(x) = x + y_\alpha(x),$$

   where α is the index chosen in the previous step.

6. Go to step 2.

Methods to generate pseudo-random numbers with a given distribution are standard knowledge in computer science and can be found in many introductory textbooks on numerics, for instance [29].

[1] Probabilities and expectations are denoted throughout by sans serif letters P and E respectively.

3.2 Applications

3.2.1 Solution of ODEs

The results obtained in subsection (2.3.2) suggest one jump for each component (i.e. $I = S$), which means Greek and Latin indices can be used interchangeably. Strictly speaking there are twice as many jumps as components (one positive and one negative for each component), but since at a fixed time for each component at most one jump has non-zero probability, it suffices to work with $I = S$. The jump in generator (2.12) can then be written as

$$x \mapsto J_\alpha(x) = x + y_\alpha(x) \qquad \text{with} \qquad y_\alpha(x) = h_\alpha(x) e_\alpha,$$

where $\alpha \in \{1, \ldots, S\}$. The h_α's, which are required to have the same sign as the right hand side Q, are called *weights* in this context. The corresponding algorithm reads as follows.

1. Fix the error control parameter(s), the stopping time $t_{\text{stop}} > 0$, set $t = 0$, choose weights $h_\alpha(x)$ and initialise the state vector according to $x = \lambda^0 \in \mathbb{R}^S$.
2. Wait an exponentially distributed time with waiting time parameter
$$\pi(x) = \sum_{\beta=1}^{S} \frac{Q_\beta(x)}{h_\beta(x)}.$$
3. If $t > t_{\text{stop}}$ then stop.
4. Choose a component index $\alpha \in \{1, \ldots, S\}$ according to the probability
$$P_\alpha(x) = \frac{Q_\alpha(x)}{h_\alpha(x)} \Big/ \pi(x).$$
5. Perform a jump according to
$$x \mapsto J_\alpha(x) = x + y_\alpha(x) = \big(x_1, \ldots, x_\alpha + h_\alpha(x), \ldots, x_S\big),$$
where α is the index chosen in the previous step.
6. Go to step 2.

Examples

In the following, a number of examples for possible choices of the weights are presented.

A) We specify the weights to be
$$h_\alpha(x) = \texttt{ATOL} \operatorname{sgn} Q_\alpha(x). \tag{3.4}$$

The constant parameter $\texttt{ATOL} > 0$ is introduced having in mind the absolute local error tolerance commonly used in the field of numerical ODE solution (see for example [17]). The waiting time parameter then becomes

$$\pi(x) = \sum_{\beta=1}^{S} \frac{Q_\beta(x)}{h_\beta(x)} = \sum_{\beta=1}^{S} |Q_\beta(x)|/\texttt{ATOL}. \tag{3.5}$$

In other words the step size is automatically adapted in the sense that, if the magnitude of the right hand side is large/small (i.e. the solution changes quickly/slowly) the time step is comparatively small/large. The component probability is

$$P_\alpha(x) = \frac{Q_\alpha(x)}{h_\alpha(x)\pi(x)} = \frac{|Q_\alpha(x)|}{\sum_{\beta=1}^{S} |Q_\beta(x)|}, \tag{3.6}$$

i.e. quantities whose (modulus of the) component of the right hand side is large are relatively likely to be chosen.

B) Another choice is given by

$$h_\alpha(x) = \texttt{ATOL} Q_\alpha(x),$$

which means the jump distance varies according to the rate of each component of the right hand side. The waiting time parameter becomes

$$\pi(x) = \sum_{\beta=1}^{S} \frac{Q_\beta(x)}{h_\beta(x)} = S/\texttt{ATOL}, \tag{3.7}$$

which means the average time step size is constant. The component probability is

$$P_\alpha(x) = \frac{Q_\alpha(x)}{h_\alpha(x)\pi(x)} = \frac{1}{S},$$

i.e. all quantities are equally likely to be chosen. Numerical experiments indicate that this method suffers from stability problems.

C) We introduce scaling functions F_α with $F_\alpha(x) > 0$ by choosing the weights to be

$$h_\alpha(x) = \frac{Q_\alpha(x)}{nF_\alpha(x)}.$$

The waiting time parameter is

$$\pi(x) = \sum_{\beta=1}^{S} \frac{Q_\beta(x)}{h_\beta(x)} = n \sum_{\beta=1}^{S} F_\beta(x)$$

and the component probability becomes

$$P_\alpha(x) = \frac{Q_\alpha(x)}{h_\alpha(x)\pi(x)} = \frac{F_\alpha(x)}{\sum_{\beta=1}^{S} F_\beta(x)}.$$

The main advantage of this method is that the computational expense is shifted from the evaluation of Q to that of F so in general one would apply this to problems with a very expensive right hand side and try to find a cheap F.

3.2.2 Direct simulation of chemical reactions

Chemical reactions have previously been modelled stochastically for instance in [13] and [20]. Consider a reaction mechanism consisting of I elementary irreversible chemical reactions

$$\nu_\alpha = (\nu_{\alpha 1}, \ldots, \nu_{\alpha S}) \quad \longrightarrow \quad \nu^*_\alpha = (\nu^*_{\alpha 1}, \ldots, \nu^*_{\alpha S}) \qquad ; \alpha \in \{1, \ldots, I\}$$

involving S different species, whose particle numbers are denoted by $x_1, \ldots, x_S$. The stoichiometric coefficients $\nu_{\alpha j}$, $\nu^*_{\alpha j}$ as well as the species x_j themselves are non-negative integers. The jump, being simply a single reaction event, is given by

$$x \mapsto J_\alpha(x) = (x_1 + \nu^*_{\alpha 1} - \nu_{\alpha 1}, \ldots, x_S + \nu^*_{\alpha S} - \nu_{\alpha S}),$$

which can be written down more concisely in the form

$$y_\alpha = \nu^*_\alpha - \nu_\alpha.$$

The rates are determined by the reaction rate functions

$$R_\alpha(x) = \gamma(x)^{1-\sum_{j=1}^S \nu_{\alpha j}} k_\alpha \prod_{j=1}^{S} \prod_{i=1}^{\nu_{\alpha j}} (x_j - i + 1), \tag{3.8}$$

where $\gamma(x)$ is a normalization factor and the k_α's are reaction rate constants.

The corresponding elementary algorithm can be enhanced in numerous ways, e.g. see subsection 3.3.2 or introduce weights h_j such that $y_{\alpha j} = h_j(\nu^*_{\alpha j} - \nu_{\alpha j})$. A method similar to the latter in a slightly different context has been proposed in [4], and other related ideas can be found in [15].

3.3 Deterministic versions of stochastic algorithms

When attempting to construct a deterministic counterpart of a stochastic algorithm, one faces the problem of how to represent the non-uniform probability distribution of the jumps. We present two slightly different ways to achieve this goal.

3.3.1 Alternative 1 - Expectation

The first step is to note that the jump index α is a random variable with probability distribution $P_\alpha = R_\alpha / \sum_\beta R_\beta$. Consequently, the jump J_α and the jump size y_α are also random variables, since they are functions of α. From this, a deterministic algorithm is obtained in the most natural way by replacing the jump and the waiting time by their expectation values:

$$\begin{array}{lcl} \text{stochastic} & & \text{deterministic} \\ x \mapsto J_\alpha(x) = x + y_\alpha(x) & \Rightarrow & x \mapsto \mathsf{E}J_\alpha(x) = x + \mathsf{E}y_\alpha(x) \\ t \mapsto t + \tau & & t \mapsto t + \mathsf{E}\tau \end{array} \tag{3.9}$$

In other words we introduce a deterministic scheme

$$\begin{array}{l} x \mapsto x + \Delta x \in \mathbb{R}^S \\ t \mapsto t + \Delta t \in [0, \infty) \end{array} \quad \text{with} \quad \begin{array}{l} \Delta x := \mathsf{E} y_\alpha \\ \Delta t := \mathsf{E}\tau \end{array}. \tag{3.10}$$

The expected jump is simply the sum of the jumps weighted by selection probabilities:

$$\mathsf{E} J_\alpha = \sum_{\alpha=1}^{I} J_\alpha(x) P_\alpha = \frac{\sum_\alpha J_\alpha(x) R_\alpha}{\sum_\beta R_\beta}$$

or equivalently

$$\mathsf{E} y_\alpha = \sum_{\alpha=1}^{I} y_\alpha(x) P_\alpha = \frac{\sum_\alpha y_\alpha(x) R_\alpha}{\sum_\beta R_\beta} = \frac{1}{\pi} \sum_\alpha y_\alpha(x) R_\alpha, \tag{3.11}$$

where (3.1) has been used in the last step. The exponential distribution (3.2) of the waiting time τ implies

$$\mathsf{E}\tau = \frac{1}{\pi}. \tag{3.12}$$

Combining this with (3.11) and applying the definitions in (3.10) one obtains

$$\frac{\Delta x}{\Delta t} = \sum_{\alpha=1}^{I} y_\alpha R_\alpha, \tag{3.13}$$

which can be used to derive the limiting equations of a given process.

Examples

A) Applying the above procedure to example A of subsection 3.2.1 simply yields Euler's algorithm as follows. Using weights (3.4), the jump can be written as

$$y_\alpha = h_\alpha e_\alpha = (\mathtt{ATOL} \operatorname{sgn} Q_\alpha) e_\alpha, \tag{3.14}$$

which implies for the rates

$$R_\alpha = \frac{Q_\alpha}{(y_\alpha)_\alpha} = \frac{Q_\alpha}{h_\alpha} = \frac{|Q_\alpha|}{\mathtt{ATOL}}.$$

Note that the time step (cf. (3.5))

$$\Delta t = \mathsf{E}\tau = \frac{1}{\pi} = \frac{1}{\sum_{\beta=1}^{S} R_\beta} = \frac{\mathtt{ATOL}}{\sum_\beta |Q_\beta|} \tag{3.15}$$

is dictated here by the theory and, in contrast to conventional adaption techniques, does not include direct error-estimation. For the expectation of the jump one obtains

$$\mathsf{E} y_\alpha = \frac{\sum_\alpha y_\alpha R_\alpha}{\sum_\beta R_\beta} = \frac{\sum_\alpha (\mathtt{ATOL} \operatorname{sgn} Q_\alpha) e_\alpha |Q_\alpha| / \mathtt{ATOL}}{\sum_\beta |Q_\beta| / \mathtt{ATOL}} = \frac{\mathtt{ATOL}}{\sum_\beta |Q_\beta|} Q = \Delta t Q.$$

The resulting propagation step

$$x \mapsto \mathsf{E}J_\alpha(x) = x + \mathsf{E}y_\alpha(x) = x + \Delta t Q$$

is precisely the standard explicit Euler method except for the automatic time adaptation. As a consequence of this step adaption, the magnitude of the changes of the components is bounded: $|\Delta x_j| = |\Delta t Q_j| \leqslant \texttt{ATOL}$. The selection probability of the components is represented here in the fact that each jump size ATOL is multiplied by its associated probability $|Q_j| / \sum |Q_i|$, which, adapting Δt according to (3.15), gives the change $\Delta x = \Delta t Q$.

An analogous calculation can be performed for example B of subsection 3.2.1 (weights $h_\alpha = \texttt{ATOL} Q_\alpha$), which also yields Euler's algorithm, i.e. $\Delta x = \Delta t Q$, but with constant time steps $\Delta t = \texttt{ATOL}/S$ (cf. (3.7)).

B) The deterministic methods obtained from simple stochastic schemes via the recipe introduced above are usually well-known explicit algorithms. Since the latter, such as Euler's method, are known to perform poorly for stiff systems, this raises the question why the stochastic algorithms are found to be relatively efficient (see section 4). Experience suggests that the concept of changing only one component of the approximate solution in each step is the reason for the robustness of the method. The following prescription imitates this feature.

1. Fix $\texttt{ATOL} > 0$, a stopping time $t_{\text{stop}} > 0$, set $t = 0$, initialise the solution vector $x = \lambda^0$ and the change vector $\Delta x = 0$.
2. Update $t \mapsto t + \Delta t$ with $\Delta t = 1 / \sum R_\beta$.
3. If $t > t_{\text{stop}}$ then stop.
4. Update the change vector

$$\Delta x \mapsto \Delta x + \mathsf{E}y_\alpha = \Delta x + \frac{\sum y_\alpha(x) R_\alpha}{\sum R_\beta}.$$

5. For each $j \in \{1, \ldots, S\}$, if $|\Delta x_j| \geqslant \texttt{ATOL}$ then update

$$x_j \mapsto x_j + \texttt{ATOL} \operatorname{sgn} \Delta x_j$$
$$\Delta x_j \mapsto \Delta x_j - \texttt{ATOL} \operatorname{sgn} \Delta x_j.$$

6. Go to step 2.

For the case of the explicit Euler method with automatic time step adaption (jump (3.14)), this algorithm is used for comparison in the numerical experiments in section 4 and is examined in more detail in [25]. The fifth step of this algorithm, which could evidently be added to every conventional solver, turns out to substantially improve its robustness.

C) When constructing a deterministic version of the second order algorithm, one obtains a method very similar to the Runge-Kutta one. The right hand side Q is replaced by some weighted average of Q evaluated at different points.

3.3.2 Alternative 2 - Average

This approach is based on the idea that on average, after a large number of steps N, the α-th jump will have occurred approximately $NR_\alpha / \sum R_\beta$ times, assuming constant rates, which is reasonable for small time steps. Therefore, we accumulate the probabilities $P_\alpha = R_\alpha / \sum R_\beta$ until they surpass unity, and whenever this is the case, the corresponding jump is performed.

1. Fix the error control parameter(s), a stopping time $t_{\text{stop}} > 0$, set $t = 0$, initialise the solution vector $x = \lambda^0 \in \mathbb{R}^S$ and the change vector $\Delta P = 0 \in \mathbb{R}^I$.
2. Update $t \mapsto t + \Delta t$ with $\Delta t = 1 / \sum R_\beta$.
3. If $t > t_{\text{stop}}$ then stop.
4. Update the change vector
$$\Delta P \mapsto \Delta P + \frac{R}{\sum R_\beta}.$$
5. For each $\alpha \in \{1, \ldots, I\}$, if $\Delta P_\alpha \geqslant 1$ then update
$$\begin{aligned} x &\mapsto J_\alpha(x) = x + y_\alpha(x) \\ \Delta P_\alpha &\mapsto \Delta P_\alpha - 1. \end{aligned}$$
6. Go to step 2.

Example

As a slightly generalized application, consider the following "hybrid" method for combined forward/reverse chemical reactions. Analogously to subsection 3.2.2, consider a chemical reaction mechanism containing I reversible reactions. In this case, there are two types of jumps, namely forward and reverse ones corresponding to single forward/reverse reaction steps respectively. The forward/reverse rates are denoted by $Q_{\alpha,\text{f}}$ and $Q_{\alpha,\text{r}}$ respectively, each given by expression (3.8) with forward/reverse rate constants. Then we have I jumps J_α with (now no longer necessarily positive) rates $R_\alpha = Q_{\alpha,\text{f}} - Q_{\alpha,\text{r}}$ such that the forward $(x + y_\alpha)$ or reverse $(x - y_\alpha)$ reaction is performed whenever R_α is positive or negative respectively.

This description leads to the following deterministic algorithm, where the role of the error control parameter `ATOL` is played now by the total particle number n.

1. Fix $n > 0$, a stopping time $t_{\text{stop}} > 0$, initialise the solution vector $x = \lambda^0 \in \mathbb{R}^S$ and the change vector $\Delta P = 0 \in \mathbb{R}^I$.
2. Update $t \mapsto t + \Delta t$ with $\Delta t = 1 / \sum |R_\beta|$.
3. If $t > t_{\text{stop}}$ then stop.

4. Update the change vector

$$\Delta P \mapsto \Delta P + \frac{R}{\sum |R_\beta|}.$$

5. For each $\alpha \in \{1, \ldots, I\}$, if $|\Delta P_\alpha| \geqslant 1$ then update

$$\begin{aligned} x &\mapsto x + y_\alpha(x) \operatorname{sgn} \Delta P_\alpha \\ \Delta P_\alpha &\mapsto \Delta P_\alpha - \operatorname{sgn} \Delta P_\alpha. \end{aligned}$$

6. Go to step 2.

An advantage of this method is that compared to conventional direct simulation techniques fewer events occur because forward and reverse reaction steps can compensate such that they are not actually performed, thereby saving computational effort. Also, since only the jumps themselves are needed, this method is preferable if the expectation of the jumps is not readily available.

This method is examined numerically in detail in [], further improved by various approximations to speed up the evaluation of the rates.

3.4 Stochastic versions of deterministic algorithms

In this subsection, we adopt a point of view opposite to the previous one, namely stochastic versions of conventional algorithms, illustrated by a number of examples. The main motivation to transform a given deterministic scheme into a stochastic one is provided by the fact that in some cases, statistical fluctuations constitute essential features of the system not visible at the deterministic level. A stochastic version can potentially exhibit additional properties, which can be used for modelling purposes, in order to describe the actual system more accurately (for example [2]).

Transferring experience from the deterministic solution of ODEs to stochastic methods has been suggested for chemical kinetics by [15].

Given a conventional (deterministic) solver of a system of ODEs. Most generally speaking, a deterministic algorithm calculates in each step the change $\Delta x \in \mathbb{R}^S$ (as a function of the right hand side of the system) in all quantities $x \in \mathbb{R}^S$ together with a (usually adaptive) step size Δt and propagates the approximate solution according to

$$\begin{aligned} x &\mapsto x + \Delta x \in \mathbb{R}^S \\ t &\mapsto t + \Delta t \in [0, \infty). \end{aligned}$$

We now define a stochastic algorithm specified by generator (2.3) to be a stochastic version of the deterministic scheme just described if the expectation of the jump process reproduces the deterministic method. In detail, we seek I, y_α and R_α such that

$$\begin{aligned} \mathsf{E} y_\alpha &= \Delta x \\ \mathsf{E}\tau &= \Delta t \end{aligned}$$

is satisfied (cf. (3.9) and (3.10)). Analogously to (3.13), this leads to the restriction

$$\sum_{\beta=1}^{I} y_\beta R_\beta = \pi \Delta x \tag{3.16}$$

imposed on the jumps and the rates. In addition, the rates are required to fulfill the condition

$$\Delta t = \mathsf{E}\tau \overset{3.12}{=} \pi^{-1} \overset{3.1}{=} \left(\sum_{\beta=1}^{I} R_\beta \right)^{-1}. \tag{3.17}$$

Due to the arbitrariness of the set of jumps, there exists a whole class of stochastic processes whose elements all satisfy the above conditions and are therefore stochastic versions of the given deterministic algorithm.

Note in particular that performing direct stochastic simulation of chemical reactions and solving the reaction rate ODEs stochastically are simply two different choices of the jumps. Therefore, one can say that chemical reactions provide a "natural" set of jumps.

Examples

A) As an example for how any solver can be interpreted as "stochastic" algorithm, we simply notice that the Euler method can be regarded as Markov process characterized by the generator

$$(\mathcal{A}\Phi)(x) = \frac{1}{\Delta t}\big[\Phi\big(x + \Delta t Q(x)\big) - \Phi(x)\big] \qquad ; x \in \mathbb{R}^S.$$

But since all quantities are propagated in each jump there is no randomness (apart from the waiting time). Evidently, *all* ODE-solvers can be regarded as stochastic processes by the trivial choice $I = 1$.

B) As shown in subsection 3.3, the examples A and B of subsection 3.2.1 are both stochastic versions of the standard explicit Euler scheme (up to time step adaption). Example A is precisely the algorithm for which the numerical experiments in section 4 have been performed. There, the relative suitability for stiff problems and robustness of this algorithm compared to the conventional Euler method are demonstrated.

C) More generally, noting that in order to satisfy the ODE we necessarily need asymptotically

$$\Delta x \sim \Delta t \cdot Q \tag{3.18}$$

for $\Delta t \to 0$, we consider – motivated by (2.12) – the generator

$$(\mathcal{A}\Phi)(x) = \sum_{\alpha=1}^{S} \frac{\Delta x_\alpha}{\Delta t h_\alpha}\big[\Phi(x + h_\alpha e_\alpha) - \Phi(x)\big],$$

for some choice of h_α consistent with (3.16).

As an example, consider the classical explicit 4th-order[2] method by Runge and Kutta (for a definition, see e.g. [29]), where the changes Δx are given by

$$\Delta x = \frac{1}{6}\big(k_1 + 2k_2 + 2k_3 + k_4\big)\Delta t$$

with

$$k_1 = Q(x), \quad k_2 = Q\Big(x + \frac{\Delta t}{2}k_1\Big), \quad k_3 = Q\Big(x + \frac{\Delta t}{2}k_2\Big), \quad k_4 = Q(x + \Delta t k_3),$$

which of course satisfies (3.18).

4 Numerical experiments

4.1 The ODE systems

Table 1: *Considered reaction mechanisms.*

Name	Species	Reactions	Reference	Website
hydrogen	10	27	[24]	www-cms.llnl.gov/combustion/combustion2.html
CO	13	35	[33]	www.galcit.caltech.edu/EDL/mechanisms/library/library.html
methane	34	164	[32]	[a]
n-heptane	107	808	[7]	[a]
iso-octane	857	3606	[8]	www-cms.llnl.gov/combustion/combustion2.html
n-decane	1218	4825	[16]	www.ensic.u-nancy.fr/DCPR/Anglais/GCR/softwares.htm

[a] The corresponding websites are no longer active. Contact the authors of the references or of this article in order to obtain the Chemkin source files.

As test systems for our algorithms we chose a number of examples for a model describing homogeneous gas-phase reaction processes at constant pressure and temperature.

The considered system of ODEs is given in each case by the reaction rate equations

$$\frac{d[X_i]}{dt} = \sum_{\alpha=1}^{I}(\nu^*_{\alpha i} - \nu_{\alpha i})\Big(\sum_{j=1}^{S} B_{\alpha j}[X_j]\Big)\Big(k_{\alpha,\mathrm{f}}\prod_{j=1}^{S}[X_j]^{\nu_{\alpha j}} - k_{\alpha,\mathrm{r}}\prod_{j=1}^{S}[X_j]^{\nu^*_{\alpha j}}\Big) \tag{4.1}$$

with $i \in \{1, \ldots, S\}$, so the number of equations simply equals the number of species (one equation for each molar concentration). The right hand side is simply the molar production rate which can be written as a summation of the rate of progress variables

[2]Note that this order is very different from the order we refer to in the context of the approximation of the derivative in the expressions for generators.

Table 2: *Initial conditions for the test systems.*

Mechanism	Species	Mole fraction	Temp. [K]
hydrogen	H_2	0.29728	1200
	O_2	0.14864	
	N_2	0.55408	
CO	CO	0.26755	1750
	H_2O	0.10000	
	O_2	0.13378	
	N_2	0.49867	
methane	CH_4	0.09564	1800
	O_2	0.19129	
	N_2	0.71307	
n-heptane	nC_7H_{16}	0.01870	1500
	O_2	0.20610	
	N_2	0.77520	
iso-octane	iC_8H_{18}	0.01664	1500
	O_2	0.20800	
	N_2	0.77536	
n-decane	$nC_{10}H_{22}$	0.01346	1500
	O_2	0.20867	
	N_2	0.77787	

for all reactions involving the i-th species. $[X_i]$ denotes the molar concentration of species i, $k_{\alpha,\mathrm{f}}$ and $k_{\alpha,\mathrm{r}}$ are the forward and reverse reaction rate constants for reaction α and $B_{\alpha j}$ accounts for third-body reactions. This notation essentially agrees with the one used in [18].

In this paper, we decided to keep temperature constant, because its inclusion as a dependent variable would simply add one equation to the system, which, apart from introducing (even more) stiffness, would not make a conceptual difference from the numerical point of view. Note, however, that one would have to choose an appropriate `ATOL`-vector, because if the same `ATOL` were used for all components virtually all jumps would occur in the temperature component and only few in the remaining ones.

Table 1 lists the considered test mechanisms including their number of species, number of reactions and references. For all systems, we used an approximately stoichiometric fuel/air ratio as initial conditions, and for the CO-combustion, we added some water. The precise values together with the temperatures are given in table 2. In each case, we set the pressure equal to one physical atmosphere ($p = 1.01325 \times 10^5$ Pa).

4.2 The considered algorithms

In our numerical experiments, we considered the following algorithm, which will be referred to as "the stochastic algorithm" later on. It is simply the first order algorithm presented in example A of subsection 3.2.1, which uses equation (3.4) for the weights, (3.5) for the waiting time parameter and (3.6) for the selection probabilities. Recall that in this version, the jump size is `ATOL` for all components and all times, whereas the time step is adapted according to the magnitude of the right hand side.

1. Fix $\texttt{ATOL} > 0$, a stopping time $t_{\text{stop}} > 0$, set $t = 0$ and initialise the state vector according to $x = \lambda^0 \in \mathbb{R}^S$.
2. Wait an exponentially distributed time with waiting time parameter
$$\pi(x) = \sum_{\beta=1}^{S} |Q_\beta(x)|/\texttt{ATOL}.$$
3. If $t > t_{\text{stop}}$ then stop.
4. Choose a component index α according to the probability
$$P_\alpha(x) = \frac{|Q_\alpha(x)|}{\sum_{\beta=1}^{S} |Q_\beta(x)|}.$$
5. Perform a jump according to
$$x \mapsto J_\alpha(x) = \big(x_1, \ldots, x_\alpha + \texttt{ATOL} \operatorname{sgn} Q_\alpha(x), \ldots, x_S\big),$$
where α is the index chosen in the previous step.
6. Go to step 2.

For comparison, we also considered the deterministic algorithm given in example B of subsection 3.3 with $h_\alpha = \texttt{ATOL} \operatorname{sgn} Q_\alpha$ (which is the algorithm proposed in [25]):

1. Fix $\texttt{ATOL} > 0$, a stopping time $t_{\text{stop}} > 0$, set $t = 0$, initialise the solution vector $x = \lambda^0$ and the change vector $\Delta x = 0$.
2. Update $t \mapsto t + \Delta t$ with $\Delta t = \texttt{ATOL}/\sum |Q_\beta|$.
3. If $t > t_{\text{stop}}$ then stop.
4. Update the change vector
$$\Delta x \mapsto \Delta x + \mathsf{E} y_\alpha = \Delta x + \Delta t Q = \Delta x + \frac{\texttt{ATOL}}{\sum |Q_\beta|} Q.$$
5. For each $j \in \{1, \ldots, S\}$, if $|\Delta x_j| \geqslant \texttt{ATOL}$ then update
$$x_j \mapsto x_j + \texttt{ATOL} \operatorname{sgn} \Delta x_j$$
$$\Delta x_j \mapsto \Delta x_j - \texttt{ATOL} \operatorname{sgn} \Delta x_j.$$
6. Go to step 2.

This will be referred to as "the deterministic algorithm" later on.

4.3 Confidence intervals and error bounds

The purpose of this subsection is to give some definitions (following [20]), which are used in our numerical experiments to measure the precision of numerical solutions.

Usually, quantities of interest can be written as some function

$$\xi(t) = \Xi\big(\lambda(t)\big)$$

of the (exact) solution λ of the initial value problem. Such a function ξ is approximated in the stochastic approach by a random variable

$$\tilde{\xi}^{(n)}(t) = \Xi\big(\tilde{\lambda}^{(n)}(t)\big), \tag{4.2}$$

where $\tilde{\lambda}^{(n)}$ denotes a sample path. In this subsection, the superscript n indicates the dependance on some error control parameter (e.g. $n = 1/\texttt{ATOL}$) such that $n \to \infty$ reproduces the exact solution.

In order to estimate the magnitude of the fluctuations and to calculate a confidence interval for the approximation (4.2), L independent sample paths with different random seed are generated. The corresponding values of the random variable are denoted by $\tilde{\xi}_1^{(n)}(t), \ldots, \tilde{\xi}_L^{(n)}(t)$, from which one can deduce the empirical mean

$$\eta_1^{(n,L)}(t) := \frac{1}{L} \sum_{l=1}^{L} \tilde{\xi}_l^{(n)}(t)$$

and the empirical variance

$$\eta_2^{(n,L)}(t) := \frac{1}{L} \sum_{l=1}^{L} \Big[\tilde{\xi}_l^{(n)}(t)\Big]^2 - \Big[\eta_1^{(n,L)}(t)\Big]^2.$$

For a fixed n, the limit $L \to \infty$ does not yield the exact solution in general, which implies, since we naturally have to use a finite n, that the systematic error

$$\big|\mathsf{E}\tilde{\xi}^{(n)}(t) - \xi(t)\big|$$

is non-zero. Clearly, we also have to use a finite L, so in practice we have to work with

$$\big|\eta_1^{(n,L)}(t) - \xi(t)\big|,$$

which is now in part statistic, because of its L-dependence. Therefore, as an estimate of the absolute overall (i.e *global*) deviation of the mean from the exact solution of the ODE after the calculation has been performed (i.e. *a posteriori*) we employ

$$c_{\text{tot}}^{(n,L)} := \frac{1}{M+1} \sum_{j=0}^{M} \big|\eta_1^{(n,L)}(t_j) - \xi(t_j)\big|, \tag{4.3}$$

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

$$t_j := j \times \frac{t_{\text{stop}}}{M}. \tag{4.4}$$

Apart from the systematic error, there is of course also a statistical error at each point in time given by the difference between the empirical mean $\eta_1(t)$ and the expectation $\mathsf{E}\tilde{\xi}(t)$. As a consequence of the central limit theorem, one can show that the probability that the expectation of the random variable $\tilde{\xi}^{(n)}(t)$ lies within the confidence interval

$$\left[\eta_1^{(n,L)}(t) - c_p^{(n,L)}(t), \eta_1^{(n,L)}(t) + c_p^{(n,L)}(t)\right]$$

is asymptotically ($L \to \infty$) equal to the confidence level p, where

$$c_p^{(n,L)}(t) := a_p \sqrt{\frac{\eta_2^{(n,L)}(t)}{L}} \tag{4.5}$$

and the value of a_p can be determined from statistical tables. Hence, we use as global (*a posteriori*) estimate of the statistical error

$$c_{\text{stat}}^{(n,L)} := \max_{j \in \{0,\ldots,M\}} \left\{c_p^{(n,L)}(t_j)\right\}, \tag{4.6}$$

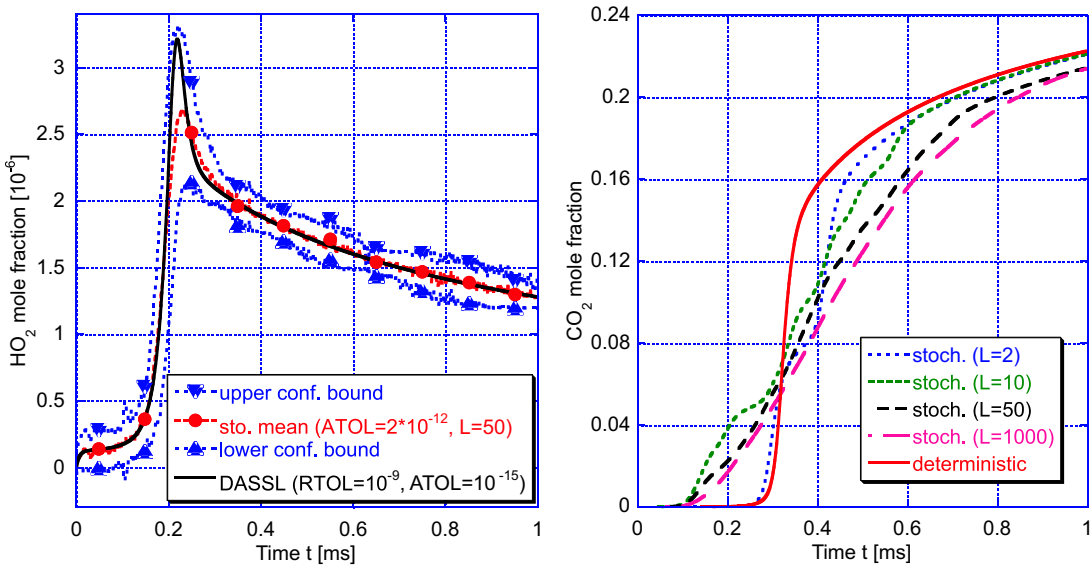
again applying splitting (4.4). Note that $c_{\text{stat}}^{(n,L)}$ depends on both n and L and consequently, because of its n-dependence, it is also systematic in nature, not purely statistic, which shows that the discrimination of systematic and statistic error is to some extent arbitrary.

4.4 Results

We created a single FORTRAN program that includes the stochastic and the deterministic algorithm as well as the state-of-the-art stiff ODE-solver DASSL (see below). The chemistry related subroutines were provided by the Sandia Chemkin package [18], which was also used to implement the mechanisms whose source files can be obtained from the websites given in table 1.

All simulations for this paper have been performed on an Intel Pentium III PC at 866 MHz running Microsoft Windows 2000. For the solution output we used $M = 2^9 = 512$ (see (4.4)) and for the error calculations a confidence level of $p = 0.999$ corresponding to $a_p \approx 3.29$ (see (4.5)). Throughout, we used $t_{\text{stop}} = 10^{-3}$ s, which was found to be suitable in the sense that on the one hand ignition takes place and on the other hand we avoid integrating over large time intervals in which the system is close to chemical equilibrium.

In order to assess how our algorithms perform compared to conventional implicit solvers, we chose the well-known and widely used solver DASSL, which is particularly suitable for stiff systems. It is based on a backward differentiation formula and designed for the solution of linearly-implicit differential-algebraic systems. For detailed technical background information, the reader may consult reference [3]. The FORTRAN source code can be obtained from the website www.engineering.ucsb.edu/%7Ecse/. In order to keep the results as comparable as possible, we set `RTOL` $= 0$ so that there remains only `ATOL` as single degree of freedom for the error control. Note that the computation time required by DASSL depends only very weakly on `ATOL` for



(a) Time dependance of the stochastic mean HO_2 mole fraction with confidence bounds (for $L = 50$ repetitions) and reference solution.

(b) Time dependance of the CO_2 mole fraction for the stochastic (for various numbers of repetitions) and deterministic algorithm ($\texttt{ATOL} = 5 \times 10^{-11}$ throughout).

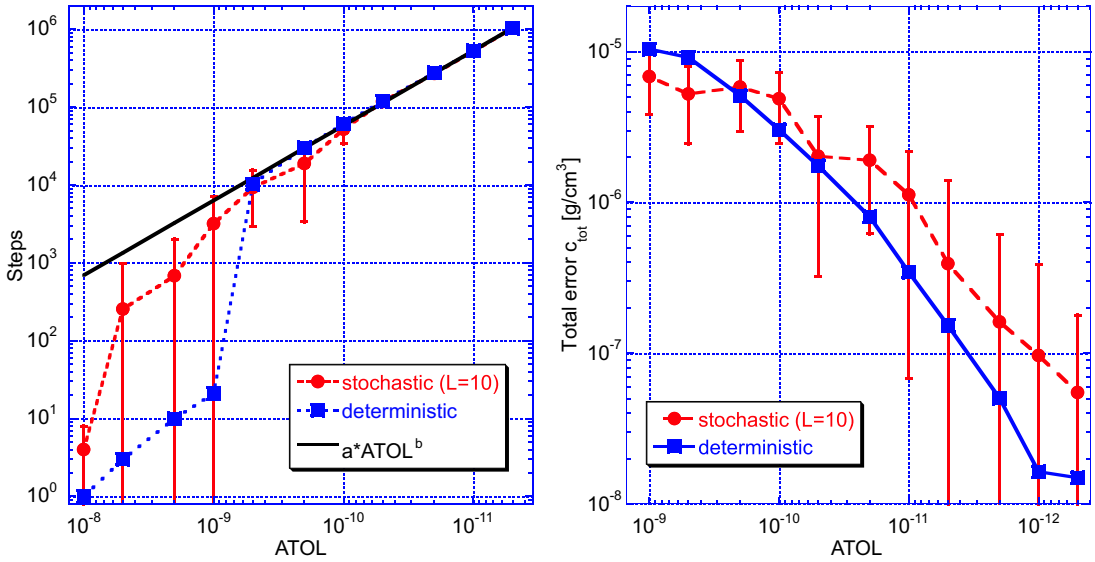
Figure 1: *Time evolution of some quantities for the CO mechanism.*

$\texttt{RTOL} = 0$. Both these parameters can be chosen scalar, because one can reasonably assume equal error tolerances for all quantities. The nature of large chemical reaction systems suggests to employ sparse methods for the manipulation of the Jacobi matrix. However, this requires quite some effort [30] and for the sake of generality, we chose not to investigate this further. We are also aware of the existence of various other methods specialised to chemical systems, but since DASSL is one of the most widespread ODE-solvers, it may serve as a benchmark and therefore facilitates the comparison to solvers developed by other authors.

The deterministic algorithm is used in this paper mainly for comparative purposes. For a more thorough investigation of its properties see [25]. Since there, the performance of the deterministic algorithm has already been compared to DASSL, we focus here on the comparison of the stochastic and the deterministic algorithm.

It turns out that stochastic algorithms are generically ill-suited to problems requiring high accuracy, mostly due to the intrinsic fluctuations. Even in the absence of the latter, as in the deterministic case, competitive efficiency is only reached in the regime of rather moderate precision [25]. But as argued in [26], for many applications, e.g. chemical kinetics coupled to fluid dynamics, such precision of the order of one percent suffices, because CFD calculations are usually similarly accurate. Therefore, we can safely take as reference solution one produced by DASSL with sufficiently small error control parameters, for instance $\texttt{RTOL} = 10^{-9}$ and $\texttt{ATOL} = 10^{-6} \times \texttt{RTOL}$. This will be referred to as "the exact solution" in the following.

As an example for the output generated by the stochastic algorithm consider the



(a) Number of steps as a function of the error tolerance `ATOL`.

(b) Total error c_{tot} of the mass density ϱ as a function of the error tolerance `ATOL`.

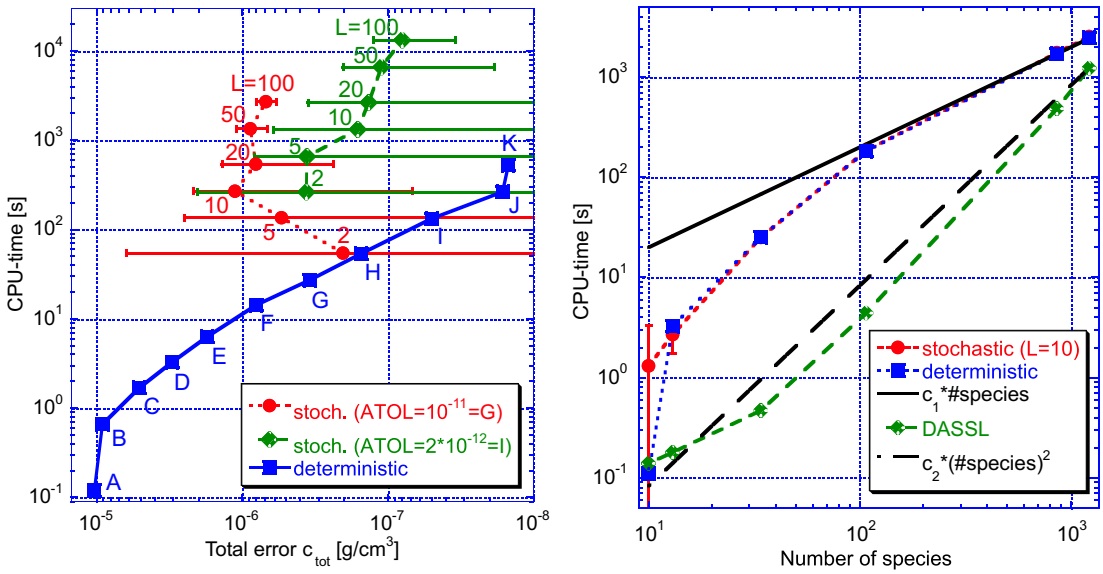
Figure 2: *Convergence of the stochastic and deterministic algorithm for the CO mechanism.*

combustion of CO (figure 1). Under the given conditions (see table 2), after roughly 0.2 ms, ignition takes place, during which the number of reaction events increases rapidly due to a chain-branching mechanism. As a consequence, the mole fractions of radicals like HO_2 reach their maximum during this period, which can be seen in figure 1(a). There, the time evolution of the HO_2 mole fraction is shown, computed using $\texttt{ATOL} = 2 \times 10^{-12}$ and $L = 50$ (number of repetitions), together with the upper and lower confidence bounds and the solution created by DASSL. As expected, the exact solution lies within the confidence interval.

Figure 1(b) displays the time evolution of the CO_2 mole fraction as calculated by the stochastic and the deterministic algorithm, where $\texttt{ATOL} = 5 \times 10^{-11}$ has been used for all curves. Four stochastic curves are shown, each with a different number of repetitions L, which indicates that the limit $L \to \infty$ does not reproduce the deterministic result. This is due to the fact that in the stochastic case the average is taken over sample paths, whereas in the deterministic case the average is taken over the jumps from a fixed state. In this particular application, a stochastic single run looks actually quite similar to the deterministic curve up to a time translation. The great variation of the ignition time is the reason for the with L increasing "flatness" of the stochastic curve.

Figure 2(a) shows for the CO mechanism the dependance of the number of steps on the error tolerance `ATOL`. The curve[3] for the stochastic algorithm is plotted per single run with statistical error bars. The number of actually occurred events represents the most immediate measure of the quality of a solution. Choosing the accuracy too

[3]In figures 2 and 3, associated points have been connected with lines as visual aid. This is not to be misunderstood as an interpolation.



(a) Computation time of the algorithms as a function of the total error c_{tot} of the mass density ϱ for the CO mechanism.

(b) Computation time of the algorithms as a function of the number of species (det./sto.: $\texttt{ATOL} = 10^{-10}$, DASSL: $\texttt{RTOL} = 0$, $\texttt{ATOL} = 10^{-10}$).

Figure 3: *Performance of the stochastic and deterministic algorithm.*

low (i.e. roughly $\texttt{ATOL} \geqslant 10^{-9}$ in this example) implies a very small number of steps and consequently a solution that may be far away from the exact solution, which is in this application reflected by the fact that the ignition does not take place. For higher accuracy (here roughly $\texttt{ATOL} \leqslant 10^{-10}$) the dependance of the number of steps on the error tolerance is given by $a\texttt{ATOL}^b$ with $a = (1.39 \pm 0.42) \times 10^{-5}$ and $b = -0.962 \pm 0.012$. This behaviour can be used to check experimentally whether the obtained solution is "sufficiently close" to the exact solution. Note also that the number of steps is directly proportional to the total CPU-time, which becomes clear from the definition of the algorithm. The number of evaluations of the right hand side of the ODE in implicit algorithms like DASSL is typically of order 10^2-10^3 for the considered systems, whereas the stochastic/deterministic algorithm needs 10^5-10^7 evaluations. But this is still small compared to conventional explicit methods, which may need 10^{12} steps or more in order to avoid severe stability problems. Looking at the structure of the stochastic/deterministic algorithm reveals that almost all the CPU-time is spent on these evaluations (at least for expensive right hand side, which is usually the case for large systems). In other words, our algorithms spend a much larger fraction of the total CPU-time on evaluations than conventional implicit algorithms. As a consequence of this, our algorithms would greatly benefit from an optimization of the evaluation of the molar production rate (the right hand side) as provided by Chemkin, whereas little impact on the conventional solvers is expected.

Both the stochastic and the deterministic method seem in general not suitable for problems requiring a large number of steps. What "large" means in a specific situation depends on the expense of the evaluation of the rates, available computational

Table 3: *Values of* `ATOL` *for figure 3(a).*

Index	`ATOL`
A	1.00×10^{-09}
B	5.00×10^{-10}
C	2.00×10^{-10}
D	1.00×10^{-10}
E	5.00×10^{-11}
F	2.00×10^{-11}
G	1.00×10^{-11}
H	5.00×10^{-12}
I	2.00×10^{-12}
J	1.00×10^{-12}
K	5.00×10^{-13}

power and the performance of alternative methods. Furthermore, this also implies that t_{stop} has to be chosen sufficiently small in order that the algorithm can produce a solution within a reasonable time span. However, for the combustion systems considered here, this does not seem to be a problem.

Figure 2(b) depicts, also for the CO mechanism, the dependence of the error c_{tot} of the mass density ϱ on the error tolerance `ATOL`, with error bars given by the statistical error c_{stat}. We chose the density because it is a function of all quantities and hence the errors c_{tot} and c_{stat} can be thought of representative for all these variables. The figure suggests that the error is roughly proportional to `ATOL`. In addition, we notice that due to the statistical fluctuations, the stochastic algorithm produces on average less accurate solutions than the deterministic one.

For the comparison of CPU-times (see figure 3(a)) we used `RTOL` $= 0$, as mentioned. The values of `ATOL` corresponding to each point in the diagram are given in table 3. Note that the statistical errors for `ATOL` $= 2 \times 10^{-12}$ are of course smaller than for `ATOL` $= 10^{-10}$, the error bars only appear larger because of the logarithmic representation. We recognize that the deterministic algorithm is more efficient than the stochastic one, even if the CPU-times were taken per single run, which is again due to the statistical fluctuations.

Figure 3(b) shows the dependence of the computation time required by each algorithm on the number of equations for fixed error control parameter `ATOL` $= 10^{-10}$. The qualitative scaling behaviour is of course independent of the actual value. Again, the stochastic curve is plotted per single run with statistical error bars. The necessity of numerous repetitions in order to obtain a sufficiently small confidence interval naturally suggests applications in which multiple runs are required anyway, such as PDF transport problems [28]. For this purpose, one can compare the single run time of the stochastic algorithm to the total run time of the other ones. The main conclusion that can be drawn from this figure is that DASSL (like other implicit solvers) scales quadratically ($c_2 = 8.33 \times 10^{-4}$ s) with the number of species whereas the stochastic and deterministic scale linearly ($c_1 = 2$ s). This strongly suggests

systems with a large number of equations as prime application of our ODE-solvers. One might argue that the conventional explicit Euler algorithm also scales linearly (because it does not make use of the Jacobi matrix of the right hand side), but not with comparable computation time, accuracy and most importantly robustness. Equal efficiency would be reached at a much higher number of equations, so that the problems would be intractable anyway (at least with currently available computational power). We deliberately avoid the term "stability" because commonly used notions (e.g. [17, 6]) seem neither to apply nor to transfer in a straightforward way to our methods.

It is also clear that the start-up costs of our algorithms are negligible, which suggest operator-splitting applications (also pointed out in [26]).

Despite their simple and explicit nature, our algorithms are robust enough to solve large stiff systems relatively efficiently.

5 Conclusion

We have given a brief introduction to how ODEs can be solved by means of (sequences of) Markov processes. Then, making use of the master equation, we have motivated our choice of Markov processes. As a consequence, we presented stochastic algorithms including several different choices of the weights, which specify these algorithms. In addition, we have indicated connections between conventional and stochastic algorithms in general and given a number of examples.

Furthermore, we studied one of our stochastic algorithms and a variant of its deterministic counterpart numerically by applying them to the combustion systems hydrogen, carbon monoxide, methane, n-heptane, iso-octane and n-decane. In addition, we compared the algorithms to one of the most widely used solver packages for stiff systems, namely DASSL.

In our numerical experiments, we have examined the dependance of the error and the number of steps on the error tolerance parameter. We have shown that the deterministic algorithm is more efficient for the considered systems than the stochastic algorithm. We also have demonstrated that the CPU-time scales linearly with the number of equations such that our algorithms can compete at moderate accuracy with conventional stiff solvers for large systems, as is the case for the combustion of n-decane for example (order 10^3 equations). In the stochastic case we have assumed multiple run applications. Last but not least, we emphasize the exceptional simplicity of our algorithms (due to their explicitness) in comparison to conventional implicit methods.

Various numerical and theoretical issues remain to be clarified, such as the stability of our algorithms for example. The negligible start-up costs together with the mentioned properties suggest as typical application large operator-splitting problems requiring moderate accuracy, such as PDF transport models. Future work includes the investigation of the deterministic direct simulation of chemical reactions and its use for flux analysis in creating skeletal mechanisms.

References

[1] M. Balthasar, F. Mauss, A. Knobel, and M. Kraft. Detailed modeling of soot formation in a partially stirred plug flow reactor. *Combustion and Flame*, 128(4):395–409, 2002.

[2] A. Bhave, M. Balthasar, M. Kraft, and F. Mauss. Measurements and simulations of homogeneous charge compression ignition combustion and emissions with exhaust gas recirculation. Technical Report 8, c4e Preprint-Series, Cambridge, 2002. (Accepted for publication in: International Journal of Engine Research).

[3] 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.

[4] H.-P. Breuer, W. Huber, and F. Petruccione. Fast Monte Carlo algorithm for nonequilibrium systems. *Phys. Rev. E.*, 53(4):4232–4235, 1996.

[5] D. L. Bunker, B. Garrett, T. Kleindienst, and G. S. Long III. Discrete simulation methods in combustion kinetics. *Combustion and Flame*, 23:373–379, 1974.

[6] 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.

[7] H. J. Curran, P. Gaffuri, W. J. Pitz, and C. K. Westbrook. A comprehensive modeling study of n-heptane oxidation. *Combustion and Flame*, 114:149–177, 1998.

[8] H. J. Curran, P. Gaffuri, W. J. Pitz, and C. K. Westbrook. A comprehensive modeling study of iso-octane oxidation. *Combustion and Flame*, 129:253–280, 2002.

[9] R. W. R. Darling. Fluid limits of pure jump markov processes: a practical guide. arXiv:math.PR/0210109, 2002.

[10] R. W. R. Darling and J. Norris. Vertex identifiability in large random hypergraphs. arXiv:math.PR/0109020, 2001.

[11] S. N. Ethier and T. G. Kurtz. *Markov Processes. Characterization and Convergence*. John Wiley & Sons, 1985.

[12] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22(4):403–434, 1976.

[13] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977.

[14] D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188:404–425, 1992.

[15] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115(4):1716–1733, 2001.

[16] 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.

[17] 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.

[18] 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.

[19] M. Kraft, P. Maigaard, F. Mauss, M. Christensen, and B. Johansson. Investigation of combustion emissions in a HCCI engine - measurements and a new computational model. *Proceedings of the Combustion Institute*, 28:1195–1201, 2002.

[20] M. Kraft and W. Wagner. Numerical study of a stochastic particle method for homogeneous gas-phase reactions. *Computers and Mathematics with Applications*, 45:329–349, 2003.

[21] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump markov processes. *J. Appl. Prob.*, 7:49–58, 1970.

[22] T. G. Kurtz. The relationship between stochastic and deterministic models for chemical reactions. *J. Chem. Phys.*, 57(7):2976–2978, 1972.

[23] P. Maigaard, F. Mauss, and M. Kraft. Homogeneous charge compression ignition engine: A simulation study on the effects of inhomogeneities. *ASME Journal of Engineering for Gas Turbines and Power*.

[24] N. Marinov, C. K. Westbrook, and W. J. Pitz. Detailed and global chemical kinetics model for hydrogen. In S. H. Chan, editor, *Transport Phenomena in Combustion*, volume 1. Taylor and Francis, Washington DC, 1996.

[25] S. Mosbach and M. Kraft. A new explicit numerical scheme for large and stiff systems of ordinary differential equations. *Jour. Sci. Comp.*

[26] 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.

[27] J. Norris. Stochastic calculus and applications. Lecture notes Part III of the Mathematical Tripos, Lent Term 2003, Cambridge, unpublished, 2003.

[28] S. B. Pope. PDF methods for turbulent reactive flows. *Prog. Energy Combust. Sci.*, 11:119–192, 1985.

[29] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++ The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2nd edition, 2002.

[30] 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.

[31] J. S. Turner. Discrete simulation methods for chemical kinetics. *J. Phys. Chem.*, 81(25):2379–2408, 1977.

[32] J. Warnatz, U. Maas, and R. W. Dibble. *Combustion*. Springer Verlag, Berlin Heidelberg New York, 1996.

[33] R. A. Yetter, F. L. Dryer, and H. Rabitz. A comprehensive reaction mechanism for carbon monoxide/hydrogen/oxygen kinetics. *Combustion Science and Technology*, 79:97–128, 1991.