Journal of Computational Physics 228 (2009) 2758–2769

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A predictor–corrector algorithm for the coupling of stiff ODEs to a particle population balance

Matthew Celnik [a], Robert Patterson [a], Markus Kraft [a,*], Wolfgang Wagner [b]

[a] Department of Chemical Engineering, Cambridge University, New Museums Site, Pembroke Street, Cambridge CB2 3RA, UK
[b] Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstraße 39, D-10117 Berlin, Germany

ARTICLE INFO

Article history:
Received 29 May 2008
Received in revised form 4 December 2008
Accepted 12 December 2008
Available online 20 January 2009

Keywords:
Predictor–corrector
Monte carlo
Population balance
Splitting
Soot

ABSTRACT

In this paper a novel predictor–corrector algorithm is presented for the solution of coupled gas-phase – particulate systems. The emphasis of this work is the study of soot formation, but the concepts can be applied to other systems. This algorithm couples a stiff ODE solver to a Monte Carlo population balance solver. Such coupling has been achieved previously for similar systems using a Strang operator splitting algorithm, however, that algorithm demonstrated several numerical issues which resulted in a high computational cost to acquire adequate precision. In particular a source-sink instability was identified whereby a large-magnitude source term present in the ODE system was competing with a similarly sized sink term in the population balance. This instability required that the splitting step size was very small in order to keep numerical error sufficiently low. A predictor–corrector algorithm has been formulated to negate this instability. An additional efficiency is gained with this algorithm as a principal computational cost of the Strang splitting algorithm is removed: the requirement to re-initialise the ODE solver every splitting step. The numerical convergence of the new algorithm is demonstrated, and its efficiency is compared to that of the Strang splitting algorithm. Substantial computation time savings are demonstrated, which allow a fixed error in three studied system functionals to be achieved with an order-of-magnitude reduction in computation time.

1. Introduction

Many physical systems involve the formation and growth of a particulate phase within a gas-phase, for example soot, titania and silica nanoparticle formation processes. The focus of this work is soot formation, but the principles can be applied to other particulate systems. The formation of soot particles and other carbon products in combustion systems is a highly complex process [1–4]. In recent years models have become available for the conversion of mass from the gas to the particulate phase [1,5–7], which calls for detailed modelling of the resulting soot particle populations.

In order to adequately characterise such systems, numerical models for the gas-phase and the particle population must be developed and combined. Generally in these systems the particulate and gas-phases are strongly coupled, as gas-phase precursors form and react with particles. Gas-phase kinetics are generally described by a finite set of ordinary differential equations (ODEs), which can be solved using well-known numerical techniques such as Runge–Kutta. Substantial effort has gone into the development of numerical population balance methods. The main numerical methods used are the method of moments with interpolative closure (MoMIC) [8], fixed [9] and moving [10] sectional methods, and Galerkin methods [11]. All of these methods explicitly reduce the evolution of the particle population to a finite set of ODEs, which can be solved

* Corresponding author.
E-mail address: mk306@cam.ac.uk (M. Kraft).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.12.030

simultaneously with the gas-phase kinetics equations. The disadvantage of these methods is that they are restricted to relatively simple particle descriptions.

Stochastic particle methods, which originate from the Bird algorithm for molecular gas dynamics [12], have also been applied to these systems [13], and in particular have been used for the mathematical study of coagulation problems [14]. These stochastic methods allow for arbitrarily complex models for individual particles [15,16] to be included with little additional impact on computation times [17] when compared to simple models, but even for simple particle models the computational requirements are much greater than for the other population methods mentioned above.

Coupling population balance calculations using the ODE methods mentioned above to chemical reaction calculations is generally possible, but computational cost can be considerable [18]. Direct coupling between stochastic particle population balance methods and the stiff differential equation solvers used to treat chemically reacting systems is a particularly challenging problem, because of the different natures of the two solution methods. Initial work in this area for soot formation avoided a direct coupling by using MoMIC, the simplest of the above methods, to provide approximations to the effects of the soot population during the solution of the gas-phase chemistry [13] and then recalculating the soot population with a stochastic particle method in a post-processing step. However, accuracy requires a close coupling, which can be achieved using an operator splitting method [19]. In that paper [19] the coupling of a stochastic soot population balance method to a standard deterministic stiff chemistry solver for a simple batch reactor was reported. The case of a perfectly-stirred reactor with outflow was also considered, but the results demonstrated that the numerical method was stretched to its limits and possibly beyond.

The purpose of this paper is to discuss the limitations of the operator splitting method [19] and to propose a more robust and less computationally demanding alternative.

This introduction is completed by a presentation of the equations describing the coupled systems already mentioned. The numerical convergence of the new method is demonstrated, and the algorithm efficiency is compared to that of the previously used Strang splitting algorithm.

1.1. Formulation of equations

At a high level a system comprising particles in a reacting gas mixture may be described by a vector $F = (f_1, f_2)^{\mathsf{T}}$, where $f_1 = (T, C_k; k = 1, 2, \ldots, K)$ contains gas-phase species concentrations and the temperature, and $f_2 = n(x)$ represents the particle population by the number density of particles of type $x \in E$. Here T is the system temperature, C_k is the concentration/fraction of species k and K denotes the number of chemical species. The fixed-length vector f_1 therefore has $K + 1$ dimensions. The number density of particles of type x is denoted by $n(x)$, where x is a vector of independent particle properties, such that $x \in \mathbb{R}^d$ for some $d \in \mathbb{N}$. The dimension d of the particle type x is left deliberately unspecified in order that the model be as general as possible. Particle models used previously include the spherical particle model ($d = 1$) [13,20], the surface-volume model ($d = 2$) [17] and the primary-particle model ($d \geqslant 2$) [21,22]. The entire particle population is then represented as a measure on the type space $E \subseteq \mathbb{R}^d$, following the approach of Eibeck and Wagner [23] and Patterson et al. [24]. At this stage no assumption is made as to whether f_1 and f_2 are discrete or continuous.

The differential equation for which an efficient solution method is sought may then be written as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} \mathcal{A}_1(f_1) \\ \mathcal{A}_2(f_1, f_2) \end{pmatrix} + \begin{pmatrix} \mathcal{B}_1(f_1, f_2) \\ \mathcal{B}_2(f_1, f_2) \end{pmatrix} \tag{1}$$

The operator subscript 1 denotes processes which occur to the gas-phase, and the subscript 2 denotes processes which affect the particle population. It is instructive to relate these operators to a physical model, hence the relationships of these operators to the physical model for a batch reactor at constant pressure are now presented, which model has been used in the subsequent numerical convergence studies. The physical interpretations of the operators in the context of a sooting system are:

(1) $\mathcal{A}_1$ is the gas-phase chemical mechanism. This has no soot particle dependence.
(2) $\mathcal{A}_2$ is the change to the particle ensemble due to gas-phase processes. For a constant pressure system this describes the expansion of gas and hence a decrease in particle number density.
(3) $\mathcal{B}_1$ is the change to the gas-phase chemistry due to particle inception and surface growth processes.
(4) $\mathcal{B}_2$ is the change to the particle ensemble due to all particle processes.

The gas-phase material balance is

$$\frac{dC_k}{dt} = \underbrace{\dot{\omega}_k(C, T) - C_k \sum_{k=1}^{K} \left(\frac{\dot{\omega}_k}{\rho}\right)}_{\mathcal{A}_1} + \underbrace{\dot{g}_k(C, n, T)}_{\mathcal{B}_1} \tag{2}$$

where $\dot{\omega}_k$ is the molar production rate of species k due to gas-phase reactions, ρ is the gas-phase density and $\dot{g}_k$ is the molar production rate of species k due to particle processes. The additional assumption of constant temperature has been made, which reduces f_1 to K dimensions. The soot particle population balance is given by

$$\frac{d}{dt}n(x) = \underbrace{-n(x)\sum_{k=1}^{K}\left(\frac{\dot{\omega}_k}{\rho}\right)}_{\mathcal{A}_2} + \underbrace{I(C,T,x) + \left(\mathcal{K}(T,x) + \sum_{p=1}^{P}\mathcal{S}_p(C,T,x)\right)n(\cdot)}_{\mathcal{B}_2} \tag{3}$$

where $I(C,T,x)$, $\mathcal{K}(T,x)$ and $\mathcal{S}_p(C,T,x)$ are the inception, coagulation and surface reaction p operators for particles of type x, respectively.

The non-linear coagulation operator $\mathcal{K}$ is defined by

$$\mathcal{K}(x)(n(\cdot)) = \frac{1}{2}\sum_{y,z\in E: y+z=x}\beta(y,z)n(y)n(z) - \sum_{y\in E}\beta(x,y)n(x)n(y) \tag{4}$$

where $\beta(x,y)$ is the coagulation kernel for the coagulation of a particle of type x and a particle of type y. The transition kernel used by Patterson et al. [20] is used for this study.

The surface reactions, which involve only a single particle at a time, are described by linear operators $\mathcal{S}_p$ defined by

$$\mathcal{S}_p(C,T,x)(n(\cdot)) = k_p(C,T,y)n(y) - k_p(C,T,x)n(x) \tag{5}$$

where $k_p(C,T,x)$ is the rate at which a particle of type x undergoes surface reaction p, and is a non-linear function of C, T and x, and where process p causes a particle of type y to become type x. For this study, the soot model of Appel et al. [1] is used to define the surface processes, which will be discussed in a subsequent section.

2. Particle–chemistry coupling strategies

As discussed above, some method is needed to allow stochastic and deterministic methods to be employed in order to solve for the soot particle population and the chemical species concentrations, respectively, in zero-dimensional systems (for example, shock tubes and plug-flow reactors).

A direct splitting of $\mathcal{A}$ and $\mathcal{B}$ was successfully used by Celnik et al. [19] to solve (1), but proved computationally expensive (around four minutes per run with 65,536 particles and a time-step size of 100 μs). Execution profiling showed that the high cost arose because a large number of small splitting steps were required to maintain adequate coupling between the $\mathcal{A}$ and $\mathcal{B}$ operators. At the start of each small step it was necessary to restart the stiff ODE solver used to calculate the effects of $\mathcal{A}$, because the solution had been changed by applying $\mathcal{B}$, and so a costly step was repeated many times.

The need for very small splitting time-steps arose because there was a numerical instability inherent in the splitting, which involved the competition of source and sink terms from the two operators. An example of this problem arose with the gas-phase species pyrene, which is produced from smaller molecules by gas-phase reactions, but is heavily consumed by the formation and growth of soot particles. The magnitudes of the pyrene source term (in $\mathcal{A}$) and the sink term (in $\mathcal{B}$) were similar and significant in relation to the size of the pyrene concentration divided by the time-step length. For long time-steps this resulted in unphysical oscillations in the value of the pyrene concentration, which became very high when operator $\mathcal{A}$ was applied and very low when operator $\mathcal{B}$ was applied. Consequently rates of change of other solution components, which depended on the pyrene concentration, could not be reliably calculated.

The system of chemical reactions represented by $\mathcal{A}$ is inevitably stiff and therefore has to be treated with implicit methods, which involve the inversion of Jacobian matrices. Such inversions are computationally very costly and so efficient solvers rely on reuse of intermediate calculation results [25]. However, the *application* of the operator $\mathcal{B}_1$, using a Monte Carlo algorithm, in a splitting step leads to a discontinuous change in $(\tilde{f}_1)$, the intermediate solution of f_1, which makes the reuse of intermediate results impossible. Therefore, after every application of $\mathcal{B}$ for the particle population part of the splitting, it was necessary to completely regenerate the internal data of the stiff ODE solver used for $\mathcal{A}$, resulting in high computational costs.

2.1. Predictor–corrector

A predictor–corrector strategy was formulated to include approximations for $\mathcal{B}_1$ in the solution to the $\mathcal{A}$ part of the splitting in order to eliminate the source-sink instability discussed previously. In setting out the algorithm below no mention is made of any internal sub-stepping that may be carried out by the solvers used for the operators $\mathcal{A}$ and $\mathcal{B}$. The iterative approach to solving split-operator problems used here is similar to the approach used by Kanney et al. [26], and the use of an approximation function for one operator is similar to the artificial sources used by Wolke and Knoth [27]. However, both those papers were concerned with the decoupling of reactive transport equations, whereas here the coupling is between the gas and particle phases.

2.1.1. Predictor

Suppose that approximate solutions to (1) have been calculated at times $t_1 < t_2 < t_3, \ldots, t_i$ to give values

$$\begin{pmatrix} f_{1,1} \\ f_{2,1} \end{pmatrix}, \begin{pmatrix} f_{1,2} \\ f_{2,2} \end{pmatrix}, \ldots, \begin{pmatrix} f_{1,i} \\ f_{2,i} \end{pmatrix}$$

respectively. The method for advancing the solution to a time $t_{i+1} > t_i$, begins with a split-predictor step:

(1) Fit a vector of functions $B^0(t)$ (for example, interpolating polynomials) defined on $[t_{i-m}, t_{i+1}]$, for $m \in \mathbb{N}$ previous points, to

$$\mathcal{B}_1(f_{1,i-m}, f_{2,i-m}), \mathcal{B}_1(f_{1,i-m+1}, f_{2,i-m+1}), \ldots, \mathcal{B}_1(f_{1,i}, f_{2,i}).$$

Note that time t_{i+1} is outside the interpolation range, hence the function extrapolates to this point.

(2) Using this approximation to $\mathcal{B}_1$, perform the first part of the splitting by solving

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\tilde{f}_1^0\\ \tilde{f}_2^0\end{pmatrix} = \begin{pmatrix}\mathcal{A}_1(\tilde{f}_1^0)\\ \mathcal{A}_2(\tilde{f}_1^0, \tilde{f}_2^0)\end{pmatrix} + \begin{pmatrix}B^0(t)\\ 0\end{pmatrix} \quad t \in [t_i, t_{i+1}] \tag{6}$$

with initial condition

$$\begin{pmatrix}\tilde{f}_1^0(t_i)\\ \tilde{f}_2^0(t_i)\end{pmatrix} = \begin{pmatrix}f_{1,i}\\ f_{2,i}\end{pmatrix}.$$

(3) Fit a vector of functions $F^0(t)$ (for example, interpolating polynomials) defined on $[t_{i-m+1}, t_{i+1}]$ to

$$f_{1,i-m+1}, f_{1,i-m+2}, \ldots, f_{1,i}, \tilde{f}_1^0(t_{i+1})$$

(4) Using this approximation to f_1, perform the second part of the splitting by solving

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{f}_2^0 = \mathcal{B}_2(F^0(t), \hat{f}_2^0) \quad t \in [t_i, t_{i+1}] \tag{7}$$

with initial condition

$$\hat{f}_2^0(t_i) = \tilde{f}_2^0(t_{i+1})$$

to get a predicted solution $(\tilde{f}_1^0(t_{i+1}), \hat{f}_2^0(t_{i+1}))^{\mathrm{T}}$ for t_{i+1}.

2.1.2. Corrector

Subsequent split-corrector iterations are performed as follows, for $J \in \mathbb{N}$ iterations:

(1) $j \leftarrow 1$

(2) Fit a vector of functions $B^j(t)$ (for example, interpolating polynomials) defined on $[t_{i-m+1}, t_{i+1}]$, for $m+1$ points (note inclusion of approximation to the point $i+1$), to

$$\mathcal{B}_1(f_{1,i-m+1}, f_{2,i-m+1}), \mathcal{B}_1(f_{1,i-m+2}, f_{2,i-m+2}), \ldots, \mathcal{B}_1(f_{1,i}, f_{2,i}), \mathcal{B}_1(\tilde{f}_1^{j-1}(t_{i+1}), \hat{f}_2^{j-1}(t_{i+1}))$$

(3) Solve

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\tilde{f}_1^j\\ \tilde{f}_2^j\end{pmatrix} = \begin{pmatrix}\mathcal{A}_1(\tilde{f}_1^j)\\ \mathcal{A}_2(\tilde{f}_1^j, \tilde{f}_2^j)\end{pmatrix} + \begin{pmatrix}B^j(t)\\ 0\end{pmatrix} \quad t \in [t_i, t_{i+1}] \tag{8}$$

with initial condition

$$\begin{pmatrix}\tilde{f}_1^j(t_i)\\ \tilde{f}_2^j(t_i)\end{pmatrix} = \begin{pmatrix}f_{1,i}\\ f_{2,i}\end{pmatrix}.$$

(4) Fit a vector of functions $F^j(t)$ (for example, interpolating polynomials) defined on $[t_{i-m+1}, t_{i+1}]$ to

$$f_{1,i-m+1}, f_{1,i-m+2}, \ldots, f_{1,i}, \tilde{f}_1^j(t_{i+1})$$

(5) Solve

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{f}_2^j = \mathcal{B}_2(F^j(t), \hat{f}_2^j) \quad t \in [t_i, t_{i+1}] \tag{9}$$

with initial condition

$$\hat{f}_2^j(t_i) = \tilde{f}_2^j(t_{i+1})$$

(6) $j \leftarrow j+1$

(7) If $j < J$ go to 2, else continue.

(8) Set solution at t_{i+1} to be last calculated values, that is,

$$\begin{pmatrix}f_{1,i+1}\\ f_{2,i+1}\end{pmatrix} \leftarrow \begin{pmatrix}\tilde{f}_1^j(t_{i+1})\\ \hat{f}_2^j(t_{i+1})\end{pmatrix}.$$

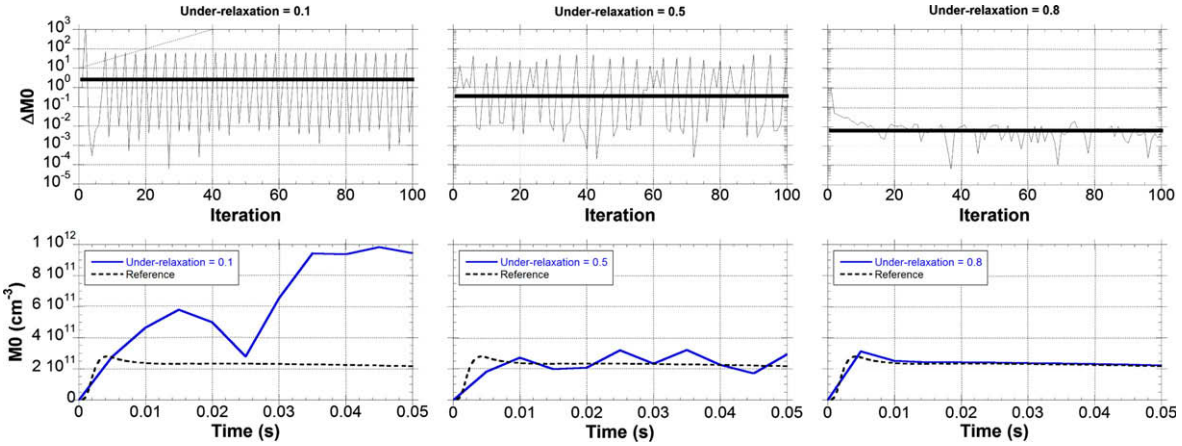

Fig. 1. Convergence with under-relaxation parameter r. Simulations were performed with $N_{max} = 65536$ and $\Delta t = 2500\,\mu s$. The top plots show the deviation from the previous iteration for 100 iterations over the final time-step. Bold horizontal lines show the statistical uncertainty calculated for that set of parameters. The bottom plots show the transient behaviour of $M0$ (zeroth moment) compared to the reference solution.

2.1.3. Under-relaxation

It was found that the above algorithm did not converge reliably for large step sizes. This problem was typically exhibited by oscillations between iterations. A standard under-relaxation method was used to enable smoothing between iterations by constructing $B^{j+1}(t)$ as explained above, but then replacing it by

$$\tilde{B}^{j+1}(t) = rB^{j}(t) + (1-r)B^{j+1}(t) \tag{10}$$

in Eq. (9). Suitable choice of the under-relaxation parameter r allows this unstable oscillation to be negated. This method is often used to prevent divergence in non-linear systems [28, p. 67] and is commonly referred to as successive over-relaxation (SOR) [29].

Fig. 1 shows results from the test case (described below) for particle number density $M0$ (zeroth moment) at different values of r. The top row of plots shows the deviation from the previous iteration over 100 iterations for the final time-step. Two points are illustrated by these plots. Firstly the nature of the unstable oscillation is clearly seen in the leftmost plot for $r = 0.1$. Secondly it is clear that the iterations very quickly converge, after 3–5 iterations there is no discernable change in the iteration deviations. The bold line gives the statistical uncertainty calculated for that simulation. The oscillation dominates over the statistical uncertainty, therefore meaningful results cannot be obtained. This finding is mirrored in the bottom row of plots, which show the transient behaviour of $M0$ compared to the reference solution. At $r = 0.1$ the profile is drastically different from the reference. Considering the middle plots at $r = 0.5$, the iteration deviations still swamp the statistical uncertainty, though the profile (bottom plot) is a better match for the reference. At $r = 0.8$ (rightmost plots) the iteration deviations are now of a similar magnitude to the statistical uncertainty and the profile is in good agreement with the reference. The value of r required to achieve a stable solution depended on the numerical parameters N_{max} and Δt, defined later. For solutions at higher numerical precision a smaller value of r could be used. For the numerical convergence study performed here it was decided to use a constant value of r for all simulations. It was found that a value of $r = 0.9$ was suitable to allow all the cases tested to converge in this manner.

2.2. Methods for the split-operators

Eqs. (6) and (8), which can be thought of as the chemical reaction part of the problem, were solved using DDASSL [30]. DDASSL is a differential/algebraic equation solver which employs backward differentiation formulae (BDF) to solve implicit systems. As DDASSL is a multi-step method it is a very stable solver, but suffers from a slow initialisation procedure. In the previous work [19] the DDASSL solver was discounted because the discontinuity in the intermediate gas-phase solution $\tilde{f}_1$ discussed earlier requires solver re-initialisation at each splitting step. Instead the RADAU5 code [31] was used. As the algorithm presented above eliminates that discontinuity in the intermediate gas-phase solution, DDASSL was used here as the more stable solver.

Eqs. (7) and (9), which contain the soot particle part of the problem, were solved using a stochastic particle method [23,13,24].

3. Numerical convergence

In the following sections the effects of the parameters of the stochastic particle algorithm and the coupling methods are reported. The main stochastic parameters were the number of computational particles, N_{max}, and the number of realisations

of the stochastic process, L. The number of computational particles, which is effectively the resolution in particle type space, E, was controlled with a particle doubling method [32,24] so that the actual particle count lay in the range $[\frac{1}{2}N_{max}, N_{max}]$, except during initial transients. One realisation of the stochastic process was used during each corrector iteration, however, for reporting results below, values are averaged over all the iterations performed for that time-step, which implies $J = L$ (see next paragraph). A spherical particle model was used to describe soot particles, whereby $E = \mathbb{N}$ and $x \in E$ is the number of carbon atoms in a particle.

The predictor–corrector algorithm introduces two additional parameters: the iteration count J and under-relaxation constant r, as described previously. Both the predictor–corrector and the Strang splitting coupling methods share a dependence on the splitting time-step size, Δt, over which coupling takes place. Multiple runs were performed to reduce the statistical uncertainty by averaging the output variables, and to give an estimation of the confidence interval. For this algorithm an additional speed increase was identified, whereby the iterations over a time-step could be used as the independent runs ($L = J$), if a copy of the solution was stored after each iteration. The solutions after each iteration were given equal weighting in this study.

3.1. Error measures

The error analysis used for this study is identical to that used for the previous study [19]. The deviation of the predicted functional values were compared to those of a high-precision reference solution $\tilde{F}$, as it would be computationally impractical to find the true solution F due to the size of the particle state space E.

3.2. Test system

The simple test case reactor used in the previous study [19] was used again for this study to allow direct comparisons of the results. The test case was a batch reactor with an initial $C_2H_2/O_2/N_2/Ar$ mixture of equivalence ratio $\Phi = 2.5$. The temperature was held constant at 1650 K and the pressure was 1 atm. No particles were present in the reactor initially and the run time was 50 ms. The soot model was taken from Appel et al. [1], which is based on the chemical mechanism development work of Wang and Frenklach [33]. In this model there are $K = 101$ chemical species, and $P = 4$ particle surface processes. The model describes a single inception event by which two gas-phase pyrene ($C_{16}H_{10}$) molecules dimerise to form a spherical particle with 32 carbon atoms.

While this test system is simple it never-the-less retains some physical basis as the conditions are similar to experimental set-ups. Therefore it is reasonable to suppose that the convergence and efficiency results obtained for this test system would be similar to those for a real combustion system.

In order to adequately discuss the numerical convergence, three functionals were chosen: particle number density, $M0$, equivalent to the zeroth moment of the particle population; soot volume fraction, Fv, equivalent to the first moment of the particle population; and the gas-phase pyrene concentration ($A4$). These functionals are identical to those used in the previous study and allow the effect of the algorithm parameters on both gas-phase and particulate properties to be discussed.

3.3. Algorithm parameters

The numerical convergence of the predictor–corrector algorithm was tested by varying the maximum particle count $N_{max} \in \{2^n, n = 6, 7, \ldots, 17, 18\}$ and by varying the time-step size Δt in the range 5—5000 μs.

The parameters of the DDASSL solver used for the stiff chemistry were those found to produced good results in earlier chemistry-only calculations, in particular the relative and absolute error tolerances were 1×10^{-3} and 1×10^{-6}, respectively. Simulations were performed with more stringent error tolerances over several orders of magnitude and no change in the convergence behaviour was observed, therefore these tolerances were deemed sufficiently small to not affect the convergence results. All predictor–corrector simulations were performed using an under-relaxation constant $r = 0.9$, which was found to give stable results for all parameters. $L = J = 100$ runs/iterations were performed for all simulations. Linear functions were used as the approximation functions $B^j(t)$ and $F^j(t)$. The number of points used to fit these functions was $m = 1$.

The reference solution $\tilde{F}$ was generated by using parameters $N_{max} = 524288$, $\Delta t = 5$ μs and $L = J = 50$. For comparison this simulation took approximately 12.5 days ($> 10^7$ s) to complete on a typical computer of today.

3.4. Convergence results

Fig. 2(left) shows the convergence of total relative error of the three functionals $M0, Fv$ and $A4$ with respect to time-step size for $N_{max} = 65536$. The trend lines in that figure show ideal first order convergence. All functionals demonstrate almost first order convergence until about $\Delta t = 100$ μs, at which point no further reduction in error appears possible. Initially it was thought that the error tolerances in the ODE solver were limiting convergence, however, the same results were obtained when the relative and absolute tolerances were changed to 1×10^{-7} and 1×10^{-10}, respectively. This suggests that the problem is unrelated to ODE solver tolerance. The study was repeated with a larger $N_{max} = 262144$, which is plotted in Fig. 2(right). This figure shows similar behaviour to the 65536 study, but limiting error is smaller. This suggests that there are two errors in competition: a splitting error associated with the time-step size, and a stochastic approximation error

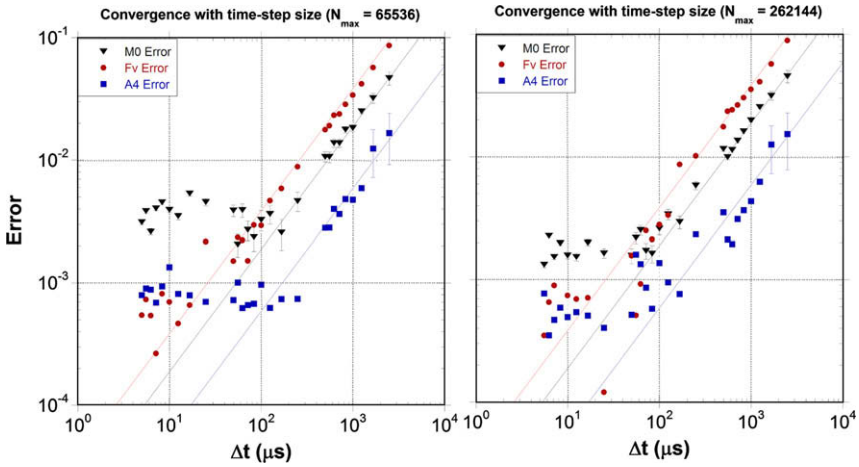

Fig. 2. Relative error convergence with step size of the predictor–corrector algorithm. The error bars show statistical uncertainty. Held constant were $N_{max} = 65536$ (left), $N_{max} = 262144$ (right) and $L = 100$ (both). Trend lines show ideal first order convergence.

controlled by the maximum particle count. The error appears to reach a limit once the time-step is made sufficiently short in Fig. 2 because it is then dominated by the stochastic approximation error.

Fig. 3 shows the convergence of total relative error with respect to maximum particle count for a fixed time-step size of 10 μs. The trend lines show order 0.5 convergence. All three functionals exhibit approximately order 0.5 convergence with respect to particle count. The very small statistical uncertainties, exhibited by small error bars, suggest that the apparent scatter of the points is due to variations in splitting error rather than the stochastic approximation. A time-step size of 10 μs was chosen because the results in Fig. 2 suggest that this is sufficiently short for total error to be controlled by N_{max}. This is backed up by Fig. 3 which demonstrates no limit of convergence in the studied range.

In the previous study [19] for Strang splitting it was found that there was approximately no dependence of total error on maximum particle count, however, this new study suggests that this was probably due to the relatively large splitting time-step of approximately 167 μs used. With that time-step size the splitting error would be dominant at all investigated particle counts. That time-step size was used because it was difficult to reduce the statistical uncertainty sufficiently to get meaningful results using the Strang splitting algorithm, which problem was not encountered using the predictor–corrector algorithm.

Fig. 4 shows the transient behaviour of particle number density in the test system for different time-step sizes. This figure illustrates that the predictor–corrector algorithm manages to approximate the reference solution far better than the Strang splitting algorithm for larger time-steps. At $\Delta t = 5000$ μs, for which only 10 steps are performed, the predictor–corrector algorithm manages to oscillate around the reference solution, whereas the Strang splitting algorithm

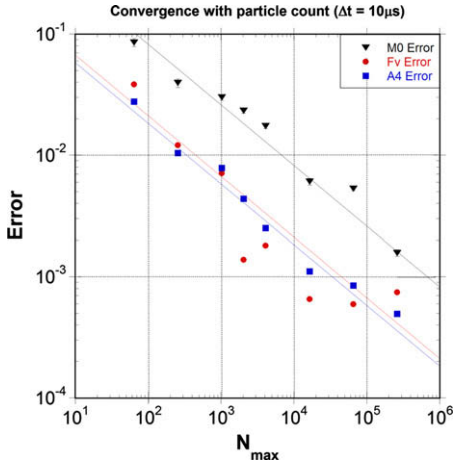

Fig. 3. Relative error convergence with maximum particle count of the predictor–corrector algorithm. The error bars show statistical uncertainty. Held constant were $\Delta t = 10$ μs and $L = 100$. Trend lines show ideal order 0.5 convergence.

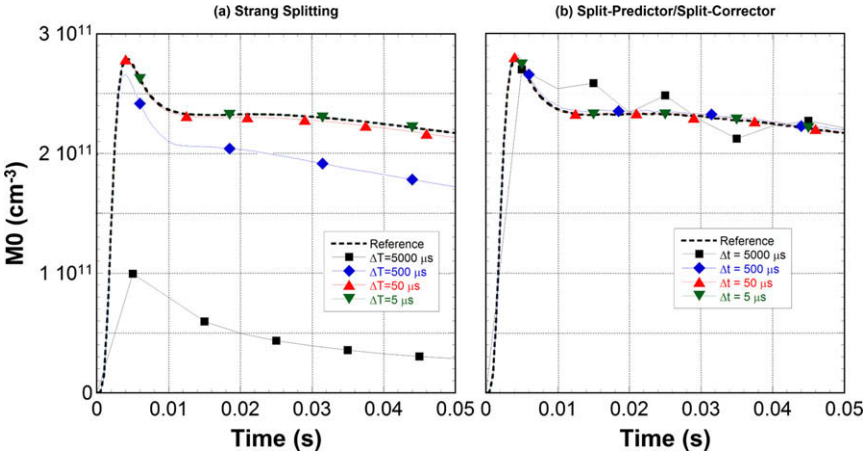

Fig. 4. Test system particle number density vs. time for Strang splitting algorithm (left) and predictor–corrector algorithm (right). Simulations were performed with $N_{max} = 65536$ and $L = 100$.

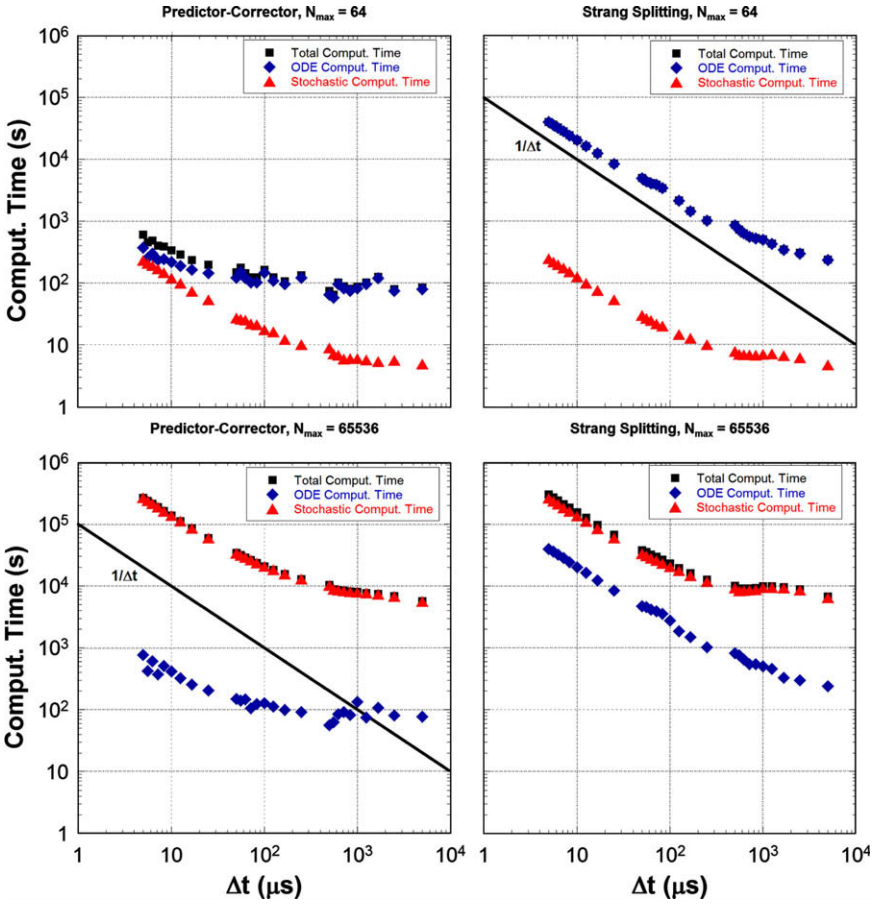

Fig. 5. Computation time as a function of time-step size for $N_{max} = 64$ (top) and $N_{max} = 65536$ (bottom). The left-hand plots show results for the predictor–corrector algorithm, and the right-hand plots show results for the Strang splitting algorithm.

hopelessly under-predicts. At $\Delta t = 500\,\mu s$ the predictor–corrector algorithm already produces an excellent fit of the reference case, while the Strang splitting algorithm has not yet reached the reference. Similar behaviour was observed for the other two functionals studied; soot volume fraction and pyrene concentration.

4. Algorithm efficiency

The discussion of computation times in this section is conducted with an understanding that such times are implementation dependent, in particular the source code compiler and computer architecture used are important. It is sufficient for this discussion to state that both algorithms were implemented in the same FORTRAN90 code using the same compiler, and that all simulations were performed on computers of the same specification. The computers were comparable to standard desktop machines available at the time.

Fig. 5 shows the dependence of computation time on the time-step size. The total computation time is further subdivided into the time spent solving the gas-phase ODE system and that spent solving the Monte Carlo stochastic particle algorithm. These plots show an approximately first order relationship between computation time and step size, which is expected. Results for both Strang splitting and predictor–corrector are plotted. For $N_{max} = 64$ (left plot) the computation time is dominated by ODE solution time, while for $N_{max} = 65536$ (right plot) the computation time is dominated by the Monte Carlo solver. This is as expected because Monte Carlo computation times generally scale approximately linearly with the number of computation particles. There is an inherent time overhead when performing a Monte Carlo step as internal variables must be set and the LPDA algorithm [17] loops over all particles at least once per step. This explains why the Monte Carlo computation time has a dependence on time-step size.

Fig. 6 shows the dependence of computation time on the maximum particle count in a similar manner to Fig. 5. The approximately first order dependence of Monte Carlo computation time on computational particle count is clearly seen. The ODE solution time is unsurprisingly not a function of computational particle count for either algorithm. These plots clearly demonstrate the computational advantage of not re-initialising the ODE solver after each time-step; the ODE solution time for the predictor–corrector algorithm (open symbols) is clearly much lower than that for the Strang splitting algorithm (solid symbols). The difference is far greater at shorter time-step sizes, approximately two orders of magnitude of computation time at $\Delta t = 5$ μs. Additionally it can be seen from Fig. 6 that this ODE step optimisation is only active for lower values

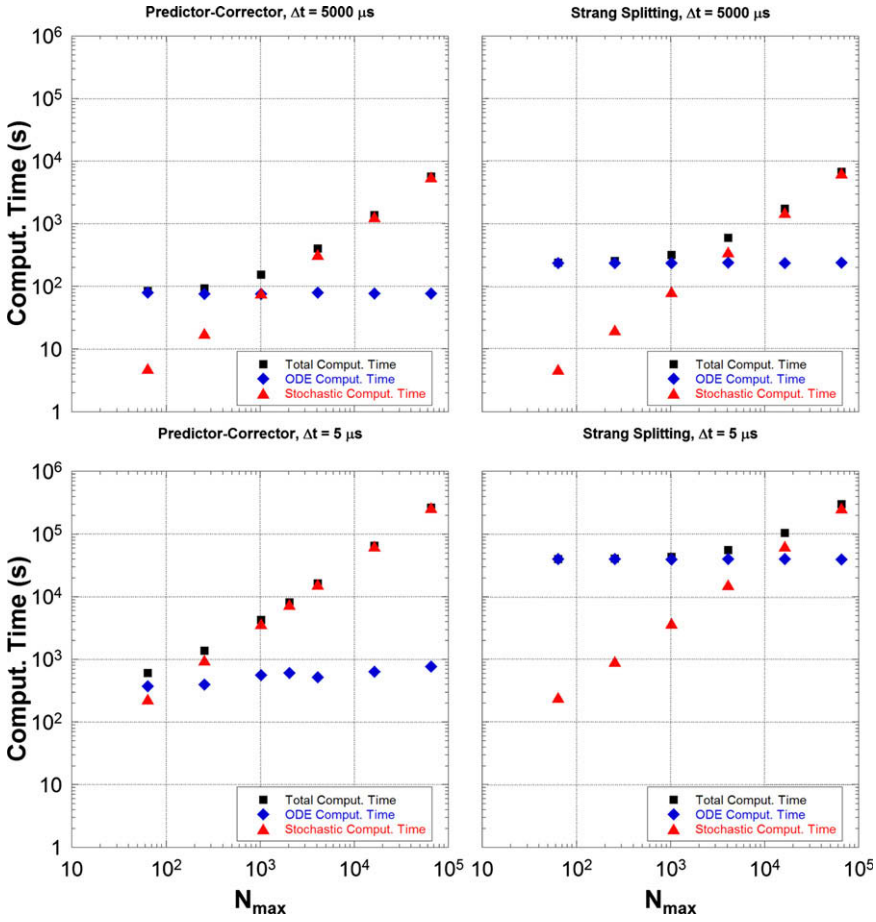

Fig. 6. Computation time as a function of maximum particle count for $\Delta t = 5000$ μs (top) and $\Delta t = 5$ μs (bottom). The left-hand plots show results for the predictor–corrector algorithm, and the right-hand plots show results for the Strang splitting algorithm.

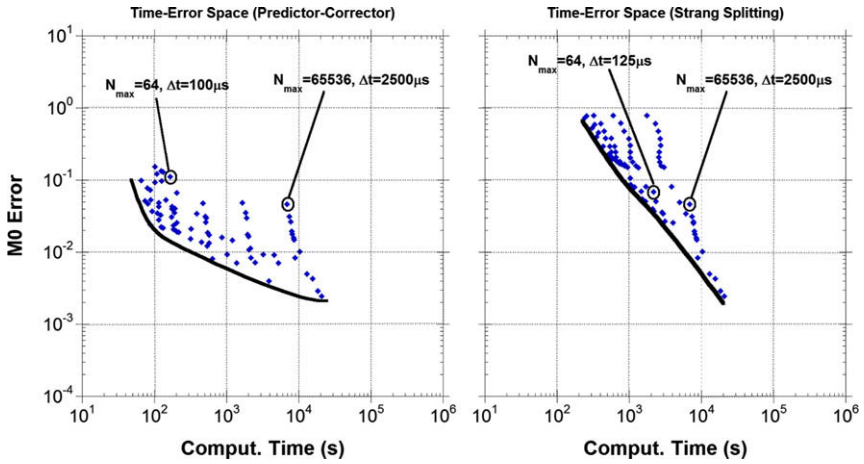

Fig. 7. Computation time–error space diagram for particle number density (zeroth moment). The point-cloud shows the error vs. computation time for each parameter set in the studied range (see text). The lines show approximately the lowest achievable error for a given computation time.

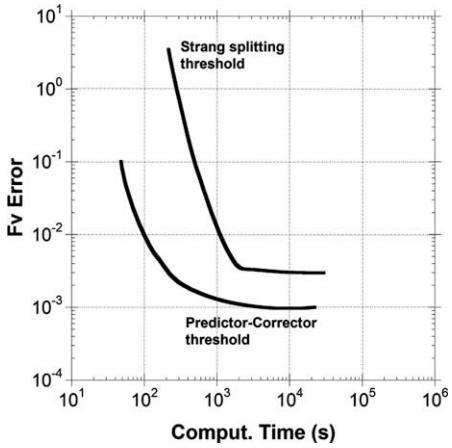

Fig. 8. Computation time–error space diagram for soot volume fraction (first moment). The lines show approximately the lowest achievable error for a given computation time. The point-clouds (as in Fig. 7) have not been shown for clarity.

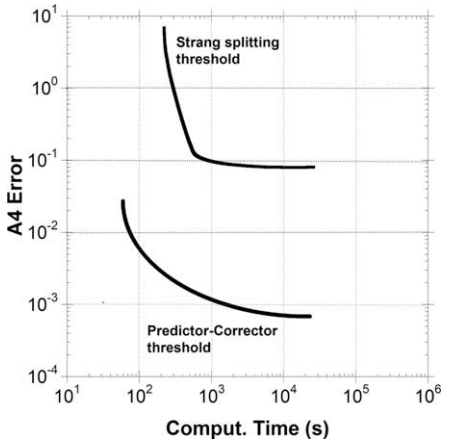

Fig. 9. Computation time–error space diagram for gas-phase pyrene concentration (*A4*). The lines show approximately the lowest achievable error for a given computation time. The point-clouds (as in Fig. 7) have not been shown for clarity.

of N_{max}; when higher N_{max} values are used, the Monte Carlo solution time dominates and the total computation times for both algorithms are similar.

Figs. 7–9 illustrate the efficiency of both algorithms by highlighting the regions in the error-computation time space occupied by the studied range of the N_{max} (64, 256, 1024, 4096, 16384, 65536) and Δt (5–100 μs) parameters. Only points for which the error is greater than the statistical uncertainty have been shown. Approximate threshold lines of lowest achievable error vs. computation time are drawn to emphasise the differences between the algorithms. It is expected that these lines are the best possible cases because any further increase in N_{max} or decrease in Δt to reduce the error can only result in a higher computation time, in accordance with the results in Figs. 5 and 6. Several features are common to all three plots. There is a general trend towards smaller errors at larger computation times, which is expected. There also appears to be a limit on the lowest possible computation time, which will be a result of fixed computational costs in the algorithms not controlled by either parameter.

In all figures it is clear that the predictor–corrector algorithm provides a significant efficiency saving over the Strang splitting algorithm. It is possible to select a fixed error of 10^{-2} (1%) with the predictor–corrector algorithm and achieve an order of magnitude reduction in computation time for functionals *M*0 and *Fv*, while the Strang splitting algorithm does not achieve that error for *A*4, though an almost two order of magnitude reduction in *A*4 error for a fixed computation time is observed. The predictor–corrector algorithm can achieve *A*4 errors below 10^{-3}, which was the relative error tolerance set in the ODE solver, while the Strang splitting algorithm cannot. The result for *A*4 is particularly encouraging, which must be due to the inclusion of an approximation to the pyrene sink term in the gas-phase ODE system. By removing the source-sink instability a very significant improvement in error is achieved.

5. Conclusions

A predictor–corrector algorithm has been developed and compared to a Strang splitting algorithm for the solution of stiff ODEs coupled to a particle population balance, solved using a stochastic particle algorithm. This new predictor–corrector algorithm incorporates an approximation to the source/sink terms for the ODE variables due to particle processes into the ODE solution. In this way the source-sink instability exhibited by Strang splitting at large splitting steps is eliminated. By removing these source/sink terms from the population balance solver a substantial computation time saving is also achieved. The predictor–corrector algorithm has been shown to converge numerically order 0.5 with maximum computational particle count (N_{max}) and first order with splitting step size (Δt). The computation times of the predictor–corrector and Strang splitting algorithms have been compared, and it has been shown that the predictor–corrector algorithm can provide substantial computation time savings, greater than one order of magnitude, depending on the choice of parameters. Finally the efficiency of the two algorithms has been compared by plotting the error-computation time space diagram for three functionals. The predictor–corrector algorithm is shown to be substantially more efficient that the Strang splitting algorithm, and can achieve the same error for an order of magnitude reduction in computation time.

References

[1] J. Appel, H. Bockhorn, M. Frenklach, Combust. Flame 121 (2000) 122.
[2] A.C. Barone, A. D'Alessio, A. D'Anna, Combust. Flame 132 (2003) 181.
[3] M. Frenklach, C.A. Schuetz, J. Ping, Proc. Combust. Inst. 30 (2005) 1389.
[4] M.J. Height, J.B. Howard, J.W. Tester, J.B.V. Sande, Carbon 42 (2004) 2295.
[5] D.F. Kronholm, J.B. Howard, Proc. Combust. Inst. 28 (2000) 2555.
[6] B. Öktem, M.P. Tolocka, B. Zhao, H. Wang, M.V. Johnston, Combust. Flame 142 (2005) 364.
[7] A. Violi, Combust. Flame 139 (2004) 279.
[8] M. Frenklach, Chem. Eng. Sci 57 (2002) 2229.
[9] H. Richter, S. Granata, W.H. Green, J.B. Howard, Proc. Combust. Inst. 30 (2005) 1397.
[10] J.Z. Wen, M.J. Thomson, S.H. Park, S.N. Rogak, M.F. Lightstone, Proc. Combust. Inst. 30 (2005) 1477.
[11] J. Appel, H. Bockhorn, M. Wulkow, Chemosphere 42 (2001) 635.
[12] G.A. Bird, Molecular Gas Dynamics, Clarendon Press, 1976.
[13] M. Balthasar, M. Kraft, Combust. Flame 133 (2003) 289.
[14] A. Eibeck, W. Wagner, SIAM J. Sci. Comput. 22 (2000) 802.
[15] M. Celnik, R. West, N. Morgan, M. Kraft, A. Moisala, J. Wen, W. Green, H. Richter, Carbon 46 (2008) 422.
[16] M.S. Celnik, A. Raj, R.H. West, R.I.A. Patterson, M. Kraft, Combust. Flame 155 (1–2) (2008) 161.
[17] R.I.A. Patterson, M. Kraft, Combust. Flame 151 (2007) 160.
[18] M.D. Smooke, C.S. McEnally, L. Pfefferle, R.J. Hall, M.B. Colket, Combust. Flame 117 (1999) 117.
[19] M.S. Celnik, R.I.A. Patterson, M. Kraft, W. Wagner, Combust. Flame 148 (2007) 158.
[20] R.I.A. Patterson, J. Singh, M. Balthasar, M. Kraft, W. Wagner, Combust. Flame 145 (2006) 638.
[21] R.H. West, M.S. Celnik, O.R. Inderwildi, M. Kraft, G.J.O. Beran, W.H. Green, Ind. Eng. Chem. Res. 46 (2007) 6147.
[22] M.S. Celnik, M. Sander, A. Raj, R.H. West, M. Kraft, Proc. Combust. Inst. 32 (2009) 639.
[23] A. Eibeck, W. Wagner, Ann. Appl. Probab. 13 (3) (2003) 845.
[24] R.I.A. Patterson, J. Singh, M. Balthasar, M. Kraft, J. Norris, SIAM J. Sci. Comput. (2006) 303.
[25] U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, Philadelphia, PA, 1998.
[26] J.F. Kanney, C.T. Miller, C.T. Kelley, Adv. Water Resour. 26 (2003) 247.
[27] R. Wolke, O. Knoth, Environ. Modell. Software 15 (2000) 711.
[28] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, vol. 24, Hemispher Publishing Corporation, New York, 1980.
[29] D.M. Young, Iterative Methods for Solving Partial Differential Equations of Elliptic Type, Ph.D. Thesis, Harvard University, 1950.
[30] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, Classics in Applied Mathematics, vol. 14, SIAM, 1995.

[31] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems, Springer Series in Computational Mathematics, second ed., vol. 14, Springer-Verlag, 1996.
[32] K.K. Sabelfeld, S.V. Rogasinsky, A.A. Kolodko, A.I. Levykin, Monte Carlo Methods Appl. 2 (1) (1996) 41.
[33] H. Wang, M. Frenklach, Combust. Flame 110 (1997) 173.