Available online at www.sciencedirect.com
ScienceDirect
Applied Numerical Mathematics 58 (2008) 319–340
APPLIED NUMERICAL MATHEMATICS
www.elsevier.com/locate/apnum

An improved and generalized second order, unconditionally positive, mass conserving integration scheme for biochemical systems

N. Broekhuizen [a,*], Graham J. Rickard [b], J. Bruggeman [c], A. Meister [d]

[a] *National Institute of Water & Atmospheric Research, PO Box 11-115, Gate 10, Silverdale road, Hamilton, New Zealand*
[b] *National Institute of Water & Atmospheric Research, Private Bag 14901, 301 Evans Bay Parade, Wellington, New Zealand*
[c] *Department of Theoretical Biology, Faculty of Earth and Life Sciences, Vrije Universiteit, De Boelelaan 1087, 1081 HV Amsterdam, The Netherlands*
[d] *Department of Mathematics and Computer Science, University of Kassel, Heinrich Plett street 40 (AVZ), D-34132 Kassel, Germany*

Available online 6 February 2007

Abstract

Bruggeman et al. [J. Bruggeman, H. Burchard, B. Kooi, B. Sommeijer, A second-order, unconditionally positive, mass-conserving integration scheme for biochemical systems, Applied Numerical Mathematics 57 (1) (2007) 36–58] presented novel first and second-order implicit integration schemes which guarantee both conservation (in a strict biochemical sense) and positive-definite results (hereafter, BBKS1 and BBKS2 respectively). In this paper we show that it is possible to achieve substantially more accurate results by making a minor modification to the BBKS-schemes (hereafter, we refer to the revised first- and second-order schemes as mBBKS1 and mBBKS2). The BBKS and the mBBKS schemes are shown to be special cases of a more generalized scheme (dubbed gBBKS). All operate by automatically slowing the forecaster time-step-averaged reaction rates in order to maintain positivity. The mBBKS scheme induces less slowing than the BBKS one. With a second modification, gBBKS-type schemes can become unusual adaptive-time-step schemes. Unfortunately, for the ODE-systems that we examined, the adaptive-mBBKS variant proves to be substantially less efficient than standard adaptive schemes (in this instance, adaptive time-step second-order explicit Runge–Kutta). Nonetheless, it is possible that the adaptive-mBBKS-scheme would become more competitive when the right-hand sides of a system of ODEs are more expensive to evaluate.
© 2006 IMACS. Published by Elsevier B.V. All rights reserved.

Keywords: Patankar-type-schemes; Runge–Kutta methods; Unconditional positivity; Mass conservation; BBKS

1. Introduction

In a recent paper by Bruggeman et al. ([2], henceforth BBKS), the mathematical implications of biochemical restrictions are clarified. The biochemical interactions under discussion require that both elemental mass and energy: (i) remain non-negative, and (ii) are neither created nor destroyed. In the mathematical domain the former requires the system to be unconditionally positive, while the latter imposes (biochemical) conservation. BBKS point out that while

* Corresponding author.
E-mail addresses: n.broekhuizen@niwa.co.nz (N. Broekhuizen), g.rickard@niwa.co.nz (G.J. Rickard), jorn.bruggeman@falw.vu.nl (J. Bruggeman), meister@mathematik.uni-kassel.de (A. Meister).

0168-9274/$30.00 © 2006 IMACS. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.apnum.2006.12.002

schemes directed at such systems exist [3,8,9] they are often limited in terms of the biochemical constraints they can usefully satisfy.

In the context of biochemical systems able to be described by systems of ODEs, BBKS demonstrate that conservation (particularly in a typical multi-nutrient system where atoms (and energy) must be conserved) requires specific relationships between the system reaction rates and the stoichiometrics to be consistently represented by the ODEs in continuous and discrete form. When combined with the need for unconditionally positive results, BBKS show that novel robust and stable solvers of such ODEs are possible.

The need for the efficient and consistent solution of biochemical ODEs is growing, particularly in attempts to model ocean biogeochemistry and its role in the global carbon cycle. Many models routinely combine several biophysical functional groups with multi-nutrient fields and their subsequent growth, decay, and transport in global ocean circulation see e.g. [4] and the references therein. Although the dynamical cores of these geophysical models contain PDE solvers, the biophysical interactions are typically updated as systems of ODEs.

These global models also stretch the available computing resources. As a result, the time increment imposed on the biophysical component comes from considerations of the efficiency of the dynamical core, and so BBKS developed their integration schemes within the framework of a predetermined time step for the biochemistry.

BBKS recognized that whilst mass-conservation is readily achieved by explicit integration schemes, these cannot usually guarantee positivity regardless of time-step. The Backward Euler scheme is conservative and (subject to certain constraints), will yield positive results, but is only first order accurate and computationally demanding for non-linear systems. Traditional, higher-order implicit schemes cannot guarantee positivity. Novel schemes [3,8] guarantee positivity at the expense of sacrificing mass- and energy conservation in certain (multiple-compound) systems.

BBKS recognized that many existing numerical integration schemes can ‘recover’ from temporary periods of (physically unrealistic) negative concentration, but argued that this is insufficient—recovery is not guaranteed. They noted that negative mass could be avoided by judicious time-step control, but situations sometimes arise in which the biological modeller is forced to work with a time-step that has been prescribed without reference to biological constraints (as in many global biogeochemical models) and in which an internal, time-splitting algorithm cannot readily be invoked to solve the biological components of the system of ODEs. Under such circumstances, the modeller is forced to adopt ad-hoc means of ensuring positivity (such as either catching negatives and resetting them to physically realistic values—perhaps with no regard to maintaining mass conservation, or through informed ‘guesses’ as to which other component(s) of the system have consumed excess material to induce the mass deficit). In their paper, BBKS introduce a formalized solution to these problems. Specifically, they develop novel first- and second-order accurate (henceforth: BBKS1 and BBKS2 respectively), fixed time-step implicit integration schemes. Their schemes guarantee both unconditional positivity and mass-conservation without resort to ad-hoc fixes and without imposing an excessive computational burden.

The primary aim of this paper is to introduce a minor change to the BBKS fixed time-step scheme to yield one that produces markedly more accurate numerical results. After introducing our modification, we show that both the original BBKS-scheme and the modified scheme are special cases of a more generalized scheme. All variants operate by introducing a ‘gradient-modifier’ term which perturbs (depresses) the time-step averaged rates of change away from those predicted by the first-(or second-)order Taylor expansion so as to guarantee positivity. Positivity is an important, but crude accuracy criterion, and we argue that all variants of the general scheme have a tendency to introduce phase errors (cf. amplitude errors) at large time-steps. We show that the magnitude of the phase error can be quantified by recognising that the so-called gradient-modifier term can instead be viewed as a time-step modifier. Recognising this, it is possible to eliminate the phase-error, and recast the scheme as an adaptive time-step method.

We also show that our modified scheme is itself simply selecting a time step out of a potential hierarchy of time step values up to and including a time step that allows one of the state variables to go exactly to zero upon integration. Our results indicate that in certain circumstances the use of values “close” to this maximum allowable time step can result in a very efficient integration scheme; however, without some kind of generalized error estimate (as in the case of adaptive schemes) use of this maximum time step can be problematic.

In Section 2, we briefly review the BBKS scheme, and in Section 3 introduce our modification. In these ensuing sections we will introduce several acronyms to refer to differing integration schemes. We define the acronyms and summarise the characteristics of each scheme in Table 1. Appendix A contains pseudo-code to illustrate how the various schemes might be implemented.

Table 1
Summary of integration schemes and acronyms used

Scheme	Acronym	Gradient modifier term	Comment
First & second order schemes from [2]	BBKSn	$\prod_{j=1}^{J} \frac{c_j^{n+1}}{c_j^n}$	
Modified BBKS-scheme	mBBKSn	$\sqrt[J]{\prod_{j=1}^{J} \frac{c_j^{n+1}}{c_j^n}}$	
Adaptive mBBKS (Appendix version, amBBKS)	amBBKS	$\sqrt[J]{\prod_{j=1}^{J} \frac{c_j^{n+1}}{c_j^n}}$	The gradient modifier term is also applied to the apparent rate of change of time, such that the realized size of the time-step is less than the nominal time-step supplied by the user. The realized time-step and end-time are communicated back to the master-stepper function of the integration routine.
Adaptive mBBKS (main-text version, samBBKS)	samBBKS	$\sqrt[J]{\prod_{j=1}^{J} \frac{c_j^{n+1}}{c_j^n}}$	Traverses the user-supplied 'external time-step' by means of one or more internal time-steps. The size of an internal time-step is chosen such that the gradient-modifier term is sufficiently close to 1.
Generalized BBKS-scheme	gBBKSn	$\sqrt[rJ]{\prod_{j=1}^{J} \frac{c_j^{n+1}}{c_j^n}}$	$r = 1/J$ implies BBKSn, $r = 1$ implies mBBKSn, $r > 1$ implies a scale-independent scheme yielding a closer approximation to Taylor expansion than mBBKSn achieves.
Explicit BBKS scheme	eBBKSn	$\min\left(1, \beta \frac{\min_{j\in J}(\frac{c_j^n}{-f_j(t^n,\vec{c}^n)})}{\Delta t}\right)$	$0 < \beta < 1$. For sufficiently small Δt this scheme is identical to the Heun-coefficient Runge–Kutta scheme; however, when that method is in danger of yielding negative concentrations, a constant gradient-reduction term is applied.

An appended integer (replacing the appended 'n') in the acronyms denotes the order of the scheme. J denotes the cardinal number of the set J^n, being the set of state-variables exhibiting a negative rate of change.

2. Methods (1). The BBKS scheme

In the context of this article, we focus on the time integration, and assume that the system of ODEs satisfy the BBKS conservation criteria [2].

The first-order BBKS scheme takes the form:

$$\vec{c}^{n+1} = \vec{c}^n + \Delta t\, \vec{f}\left(t^n, \vec{c}^n\right) m\left(\vec{f}, \vec{c}^n\right) \tag{1}$$

in which $\vec{c}^n$ denotes the vector of state variables at time-level n, Δt is the time-step, $\vec{f}(t^n, \vec{c}^n)$ is the vector of instantaneous estimated temporal-gradients for the state-variables at time-level t^n and $m(\vec{f}, \vec{c}^n)$ is a scalar modifier $(0 < m(\vec{f}, \vec{c}^n) < 1)$. In higher order schemes, a new value for the term $m(\vec{f}, \vec{c}^n)$ is calculated for each 'sub-time-step' prediction, as well as for the ultimate full-time-step 'corrector' projection. We will adopt the notation $m_l(\vec{f}, \vec{c}^n)$; $l \in \{1, 2, \ldots, L\}$ when it becomes necessary to distinguish between the sequential values in a Lth-order scheme.

Specifically, BBKS proposed that the first-order scheme (henceforth, BBKS1) be based upon a modification of the Euler integration method viz:

$$\vec{c}^{n+1} = \vec{c}^n + \Delta t\, \vec{f}\left(t^n, \vec{c}^n\right) \prod_{j\in J^n} \frac{c_j^{n+1}}{c_j^n} \tag{2}$$

with $J^n = \{i\colon\ f_i(t^n, \vec{c}^n) < 0,\ i \in \{1, \ldots, I\}\}$, where J^n represents the set of indices for state variables having a negative temporal derivative at time-level n. Thus, in the original BBKS1 scheme, $m(\vec{f}, \vec{c}^n) = \prod_{j\in J^n} \frac{c_j^{n+1}}{c_j^n}$ (Henceforth, we use the phrase 'change ratio' to refer to the ratios $(\frac{c_j^{n+1}}{c_j^n})$ and $\prod_{j\in J^n}(\frac{c_j^{n+1}}{c_j^n})$. We will rely upon context to distinguish between the simple quotient and the product-quotient variants).

Their second-order method (BBKS2) is based upon a modification of the second-order Runge–Kutta method (using the Heun coefficients) viz:

$$\vec{c}^{(1)} = \vec{c}^n + \Delta t\, \vec{f}\big(t^n, \vec{c}^n\big) \prod_{j\in J^n} \frac{c_j^{(1)}}{c_j^n}, \tag{3a}$$

$$\vec{c}^{n+1} = \vec{c}^n + \frac{\Delta t}{2}\big(\vec{f}\big(t^n, \vec{c}^n\big) + \vec{f}\big(t^{n+1}, \vec{c}^{(1)}\big)\big) \prod_{k\in K^n} \frac{c_k^{n+1}}{c_k^{(1)}}. \tag{3b}$$

J^n retains its earlier definition and $K^n = \{i\colon\ (f_i(t^n, \vec{c}^n) + f_i(t^{n+1}, \vec{c}^{(1)})) < 0,\ i \in \{1, \ldots, I\}\}$. Eq. (3a) is an implicit equation for the quantity $\vec{c}^{(1)}$—which represents a preliminary estimate the vector of state-variable concentrations at the end of the forthcoming time-step. Eq. (3b) is an implicit equation for $\vec{c}^{n+1}$ (i.e. the final, refined estimates of the state-variable values at the end of the forthcoming time-step). BBKS show that Eq. 3(b) can be rewritten in a form analogous to Eq. (2):

$$\vec{c}^{n+1} = \vec{c}^n + \frac{\Delta t}{2}\big(\vec{f}\big(t^n, \vec{c}^n\big) + \vec{f}\big(t^{n+1}, \vec{c}^{(1)}\big)\big) \prod_{k\in K^n} \frac{c_k^n}{c_k^{(1)}} \prod_{k\in K^n} \frac{c_k^{n+1}}{c_k^n}. \tag{3c}$$

BBKS demonstrated that the implicit equations (2), or (3a) & (3b) or (3c) can be solved by making the substitution $p = \prod_{j\in J^n} c_j^{n+1}/c_j^n$, yielding an implicit equation for p:

$$g(p) = \prod_{j\in J^n} (1 + a_j p) - p = 0 \tag{4a}$$

with

$$a_j = \Delta t f_j\big(t^n, \vec{c}^n\big)/c_j^n. \tag{4b}$$

For Eq. (2) to have a positive solution for the ith coupled ODE, requires:

$$p > \frac{-c_i^n}{\Delta t f_i(t^n, \vec{c}^n)}, \quad \text{if } f_i\big(t^n, \vec{c}^n\big) > 0, \tag{5a}$$

$$p < \frac{-c_i^n}{\Delta t f_i(t^n, \vec{c}^n)}, \quad \text{if } f_i\big(t^n, \vec{c}^n\big) < 0. \tag{5b}$$

Furthermore, since all of the ODEs must yield positive solutions Eq. (5a) is superseded by the requirement:

$$p = \prod_{j\in J^n} \frac{c_j^{n+1}}{c_j^n} > 0. \tag{5c}$$

The upper bound to p is given by:

$$p_{\max} = \Gamma \tag{5d}$$

where

$$\Gamma = \min\left(1, \min_{j\in J^n}\left(\frac{-c_j^n}{\Delta t f_j(t^n, \vec{c}^n)}\right)\right). \tag{5e}$$

BBKS further demonstrated that $g(p)$ has only one real root in the range $0 < p < p_{\max}$.

Thus, to solve Eq. (2), one first uses a root finding algorithm to solve Eq. (4) for p and then substitutes the resultant value into Eq. (2) in order to make the projection across the integration interval. An analogous procedure is adopted in the second-order method (Eq. (3)).

3. Methods (II). A modified fixed time-step scheme

Despite the merits of the BBKS1 and BBKS2 schemes, inspection of Eq. (2) reveals an undesirable property—it is not *scale-independent*. The gradient-modifier term ($m(\vec{f},\vec{c}^n) = \prod_{j\in J^n} \frac{c_j^{n+1}}{c_j^n}$) must become ever smaller as the number of members in the set J^n increases—even if the rates of decline in the additional members are such that they are in no imminent danger of becoming negative. To illustrate the undesirable nature of the scale-dependence, consider a sequence of reaction vessels coupled by diffusion, each having identical initial and physical conditions. Under these circumstances, the numerical solution for each and every vessel should be identical and independent of the number of vessels. Unfortunately, the schemes offered by BBKS would not satisfy the independence criterion if applied naively (i.e. by solving the ODEs for all of the vessels simultaneously).

We propose a minor modification to the gradient modifier: namely that it be given by the geometric average of the change ratio associated with each of the declining state-variables (rather than the product of these individual ratios as in the BBKS-method). Averaging removes the scale-dependence of the scheme. Adopting the geometric average ensures that the implicit part of the integration scheme remains that of finding the root of a univariate, implicit polynomial (see below)—as it is for the original BBKS schemes. With this modification, the first-order scheme is (henceforth mBBKS1):

$$\vec{c}^{n+1} = \vec{c}^n + \Delta t\,\vec{f}\big(t^n,\vec{c}^n\big)\left(\prod_{j\in J^n}\frac{c_j^{n+1}}{c_j^n}\right)^{\frac{1}{J}} = \vec{c}^n + \Delta t\,\vec{f}\big(t^n,\vec{c}^n\big)p^{\frac{1}{J}}. \tag{6}$$

In which J is used to denote the cardinal number of the set J^n. Eq. (6) can be solved in a manner analogous to that employed in the original BBKS-scheme. Since we require

$$m\big(\vec{f},\vec{c}^n\big) = \sqrt[J]{\prod_{j\in J^n}\frac{\vec{c}^{n+1}}{\vec{c}^n}} < \min_{j\in J^n}\left(\frac{-c_j^n}{\Delta t f_j(t^n,\vec{c}^n)}\right),$$

the upper limit upon p is given by:

$$p_{\max} = \min\left(1, \min_{j\in J^n}\left(\left(\frac{-c_j^n}{\Delta t f_j(t^n,\vec{c}^n)}\right)^J\right)\right), \tag{7}$$

so that the upper limit for $p^{1/J}$ the in Eq. (6) is, of course, still Γ from Eq. (5e).

The second-order equivalent of our modified scheme (mBBKS2) is:

$$\vec{c}^{(1)} = \vec{c}^n + \Delta t\,\vec{f}\big(t^n,\vec{c}^n\big)\left(\prod_{j\in J^n}\frac{c_j^{(1)}}{c_j^n}\right)^{\frac{1}{J}}, \tag{8a}$$

$$\vec{c}^{n+1} = \vec{c}^n + \frac{\Delta t}{2}\big[\vec{f}\big(t^n,\vec{c}^n\big) + \vec{f}\big(t^{n+1},\vec{c}^{(1)}\big)\big]\left(\prod_{k\in K^n}\frac{c_k^n}{c_k^{(1)}}\right)^{\frac{1}{K}}\left(\prod_{k\in K^n}\frac{c_k^{n+1}}{c_k^n}\right)^{\frac{1}{K}}. \tag{8b}$$

In which K denotes the cardinal number of the set K^n. The implicit equation for the slope-modifier is:

$$g_{\mathrm{mBBKS}}(p) = \prod_{j\in J^n}\left(1 + \frac{(\Delta t\,\vec{f}(t^n,\vec{c}^n)p^{\frac{1}{J}})}{\vec{c}^n}\right) - p. \tag{9}$$

Following the example of BBKS we can show that this polynomial declines monotonically within the relevant range of p. Differentiating with respect to p yields:

$$\frac{dg_{\mathrm{mBBKS}}(p)}{dp} = \sum_{i\in J^n}\left(\frac{p^{\frac{1}{J}-1}}{J}\frac{\Delta t f_i(t^n,\vec{c}^n)}{c_i^n}\prod_{j\in J^n,\,j\neq i}\left(1 + \frac{\Delta t f_i(t^n,\vec{c}^n)}{c_i^n}p^{\frac{1}{J}}\right)\right) - 1. \tag{10}$$

Substituting $a_i = \frac{\Delta t f_i(t^n, \vec{c}^n)}{c_i^n}$ yields:

$$\frac{dg_{\mathrm{mBBKS}}}{dp} = \sum_{i \in J^n} \left(\frac{p^{\frac{1}{J}-1}}{J} a_i \prod_{j \in J^n, j \neq i} \left(1 + a_i p^{\frac{1}{J}}\right) \right) - 1. \tag{11}$$

Within the permissible p domain, $a_i < 0$ for all $i \in J^n$ and $1 + a_i p^{\frac{1}{J}} > 0$. Thus, inside the region of interest $dg(p)/dp < 0$; $g(0) = 1$ and $g(p_{\max}) < 0$. Therefore, despite the differing form for the gradient modifier term, the associated implicit polynomial for p still declines monotonically within the range of interest and still can be solved in a manner analogous to that proposed by BBKS.

BBKS advocated a (relative) error-tolerance of 10^{-9} when using the bisection method to estimate the root p of $g(p)$ (Eq. (4)). In the BBKS1 and BBKS2 schemes the gradient-modifier term $m(\vec{f}, \vec{c}^n) = p$. Thus, they were also estimating $m(\vec{f}, \vec{c}^n)$ with this degree of precision. In the mBBKS2 scheme, however, $m(\vec{f}, \vec{c}^n)$ is a non-linear function of p whenever more than one state-variable is declining. Thus, in the mBBKS2 scheme, p may need to be located more accurately if $m(\vec{f}, \vec{c}^n)$ is to be located with an accuracy of 10^{-9}. In the simulations described below, p is first estimated to a relative error-tolerance of 10^{-9}. If necessary, it was then further refined until $m(\vec{f}, \vec{c}^n) = p^{\frac{1}{J}}$ was judged to have converged adequately (we have adopted a value of 10^{-5} in the simulations presented below; in effect, our convergence criterion is looser than the one adopted by BBKS).

It is worth noting that both the BBKS and mBBKS schemes can be viewed as special cases of a more general scheme. The first-order variant of the general scheme is:

$$\vec{c}^{n+1} = \vec{c}^n + \Delta t \cdot \vec{f}\left(t^n, \vec{c}^n\right) \left(\prod_{j \in J} \frac{c_j^{n+1}}{c_j^n} \right)^{\frac{1}{q}}. \tag{12}$$

We will refer to this as the gBBKS scheme. It can be extended to higher order in the same way as BBKS2 and mBBKS2 were derived from their first-order counter-parts. The original BBKS scheme corresponds to $q = 1$. The mBBKS scheme corresponds to $q = J$, but scale independence will be ensured for $q = rJ$. Choosing $r \geqslant 1$ ensures that the term $(\prod_{j \in J} \frac{c_j^{n+1}}{c_j^n})^{\frac{1}{q}}$ is convex with respect to the quotient (i.e. the gradient modifier will not begin to depart substantially from one until at least one of the quotients $(\frac{c_j^{n+1}}{c_j^n})$ is substantially below one). Intuitively, we anticipate that as the value of r is increased, the realized gradient-modifier term will converge (from below) upon Γ (the value beyond which positivity would be violated). Any member of the set of gBBKS-schemes operates by perturbing the time-step-averaged rate of change away from that of (the approximation to) the Taylor polynomial expansion around $[t^n, \vec{c}_n]$. In effect, the form chosen for the exponent q determines what fraction of the available space between 0 and Γ can be used by the numerical scheme. Recognizing these two facts, one might legitimately ask: "rather than going to the effort of solving an implicit polynomial to find a gradient-modifier term, why not simply calculate Γ and then set $m(\vec{f}, \vec{c}^n) = \beta\Gamma$, where $0 < \beta < 1$ (likely close to 1), is a user-prescribed quantity?". Such a scheme would guarantee positivity without incurring the cost associated with an implicit-root-finding problem. We will refer to this explicit variant as eBBKS.

We have not been able to derive a proof that gBBKS schemes will invariably yield more accurate results than an eBBKS scheme (though it is easy to imagine values of β which would cause eBBKS to be inaccurate). Instead, we have also applied the eBBKS scheme to our two test problems (described below) using $\beta = 0.9999$, $\beta = 0.99$ and $\beta = 0.90$.

4. Error estimation and implicit adaptive time-stepping

Up to this point, we have considered $m(\vec{f}, \vec{c}^n)$ to be a factor which is used in combination with the instantaneous estimates of the temporal rates of change to derive an improved approximation to the time-step averaged rate of change. This interpretation is unambiguously correct in the Patankar-like schemes [6] because, in those, the numerical solution to each ODE may adopt a different modifier term. In contrast, for gBBKS-like and eBBKS-like schemes, the numerical value for the modifier is held constant in *all* of the difference equations that generate the numerical solution. The modifier appears as a member of a triple-product $(\Delta t \vec{f}(t^n, \vec{c}^n) m(\vec{f}, \vec{c}^n))$. Thus, it is equally valid to

view the factor as a time-step modifier. BBKS recognised this fact, but did not elaborate upon its consequences. With this time-step scaling in mind it becomes clear that, in the context of a *fixed* time-step scheme, the gBBKS and eBBKS schemes operate by slowing the rate at which the dynamical system moves along its attractor. In effect, fixed-time-step BBKS-type schemes maintain positivity by slowing all of the reaction kinetics. The slowing is greatest whenever a state-variable is in imminent danger of becoming negative, but is never entirely absent. Clearly, slowing of at least some of the reaction-rates is warranted when projections based upon an explicit integration scheme would violate positivity. Nonetheless, the degree of slowing that is introduced by gBBKS scheme is determined only by a positivity requirement (rather than a requirement to derive an accurate approximation to the true, time-step averaged rates of change). For this reason, gBBKS-like schemes (including eBBKS) are prone to introducing phase errors relative to the results stemming from the corresponding fine-time-step, explicit Runge–Kutta scheme (i.e. they tend to induce excessive slowing). The phase errors will be identical for all state-variables, so may be of little concern in autonomous systems (because the shape of the attractor is preserved), but, for time-dependent systems, the phase-errors do have the potential to seriously perturb the shape of the attractor.

It is possible to quantify the magnitude of the phase-error by introducing an auxiliary state-variable t_{ds}; this state-variable represents the true-time or 'dynamical system time'. The corresponding differential equation represents the instantaneous rate of change of this time relative to that of the apparent time of the integration-system (t_{is}, being the cumulative product of the number of completed integration cycles, and the user-specified, fixed 'time-step'):

$$\frac{dt_{\mathrm{ds}}}{dt_{\mathrm{is}}} = 1.$$

As with the other differential equations, the numerical solution to this ODE involves multiplication by the factor $m(\vec{f}, \vec{c}^n)$ (note however that t_{ds} never contributes to the value of $m(\vec{f}, \vec{c}^n)$—both because it is non-conservative, and because it is monotonically increasing).

BBKS envisaged that their scheme would be embedded within a more complex (external) scheme, and would be adopted because the user had no control over the time-step of the external scheme. With our new insight we see that one can accurately quantify accumulating phase-error that arises in this situation. Furthermore, one can envisage a variety of options to regulate the rate at which the phase-error accumulates. We describe the second-order variant of one of these below (dubbing the resultant scheme samBBKS2); a second, more complex scheme is described in Appendix B. Both descriptions are rendered in terms of mBBKS2, but are equally applicable to other gBBKS2 variants, eBBKS2 and the first order variants of the three schemes.

The samBBKS2 scheme operates as follows. Upon entry to the samBBKS2 algorithm, an internal time-step (Δt_{ds}) should be defined and set equal to the external time-step Δt_{is}. A preliminary estimate of the modifier term $m_1(\vec{f}(t^n, \vec{c}^n), \vec{c}^n)$ should be made on the basis of this time-step (as in the mBBKS-scheme). If it is not sufficiently close to 1.0, this is taken as evidence that higher-order terms in the Taylor-expansion are not negligible, and that a smaller time-step is required. The internal time-step is therefore reduced (for example, halved), and $m_1(\vec{f}(t^n, \vec{c}^n), \vec{c}^n)$ re-evaluated on the basis of the revised internal time-step. Step-size reduction continues until $m_1(\vec{f}(t^n, \vec{c}^n), \vec{c}^n)$ is judged to be sufficiently close to 1.0. The algorithm then uses the mBBKS method (as initially described) to make a projection over the internal time-step. The cycle is repeated until the entire external time-step has been traversed. At the beginning of each successive internal cycle, the initial size of the forthcoming internal time-step is set to: $\Delta t^s_{\mathrm{ds}} = \Delta t_{\mathrm{is}} - \sum_{r=0}^{s-1} \Delta t^r_{\mathrm{ds}}$—where $(s-1)$ denotes the number of internal times-steps that have been completed within the present external time-step (the realized size of the forthcoming time-step will be reduced subsequently if the gradient modifier, m_1, proves too small).

5. Model problems

BBKS present results for three dynamical systems. The first is a simple, linear system:

$$\frac{dc_1}{dt} = c_2 - \alpha c_1, \qquad \frac{dc_2}{dt} = \alpha c_1 - c_2. \tag{13}$$

At any instant, only one of the two state variables can be in decline. Thus, the BBKS and mBBKS gradient-modifier terms are numerically identical, and the two schemes yield identical results (not shown).

The first non-linear system adopted by BBKS (their 'Simple Non-Linear System') represents a (highly simplified) nutrient/phytoplankton/detritus system. The phytoplankton (P) grow under the constraining influence of two nutrients

(C and N in BBKS, but N_1 and N_2 within this m.s. to preclude confusion with the elemental symbols for carbon and nitrogen—which are discussed later). When a phytoplankter dies, it passes into an inert detrital pool (D). The phytoplankton population maintains a fixed internal $N_1 : N_2$ ratio.

$$\frac{dN_1}{dt} = -ar_{\max}\frac{N_1}{K_{N_1}+N_1}\frac{N_2}{K_{N_2}+N_2}P, \tag{14a}$$

$$\frac{dN_2}{dt} = -br_{\max}\frac{N_1}{K_{N_1}+N_1}\frac{N_2}{K_{N_2}+N_2}P, \tag{14b}$$

$$\frac{dP}{dt} = r_{\max}\frac{N_1}{K_{N_1}+N_1}\frac{N_2}{K_{N_2}+N_2}P - eP, \tag{14c}$$

$$\frac{dD}{dt} = eP \tag{14d}$$

with $a = b = 1$, $K_{N_1} = K_{N_2} = 1$ (ML^{-3}), $r_{\max} = 1$ (T^{-1}) and $e = 0.3$ (T^{-1}), and initial conditions $N_1 = 29.98$, $N_2 = 9.98$, $P = D = 0.01$.

Given the initial conditions, N_1 and N_2 decline asymptotically, P initially rises and subsequently falls and D rises asymptotically. Thus, the system has either two, or three declining state-variables.

The second non-linear dynamical system examined by BBKS is the Robertson test case:

$$\frac{dY_1}{dt} = -0.04Y_1 + 10^4 Y_2 Y_3, \tag{15a}$$

$$\frac{dY_2}{dt} = 0.04Y_1 - 10^4 Y_2 Y_3 - 3\times 10^7 Y_2^2, \tag{15b}$$

$$\frac{dY_3}{dt} = 3\times 10^7 Y_2^2 \tag{15c}$$

with initial conditions: ($Y_1 = 1.0$, $Y_2 = Y_3 = \varepsilon$, where ε is the smallest number greater than zero that is representable in double-precision);

Though not made explicit in their paper, BBKS endeavoured to solve this system of equations using a time-step which increased exponentially with respect to simulated time. BBKS do not present detailed simulation results but they do note that their second order scheme 'stalled the system through extremely small p-factors' (i.e. gradient-modifier terms).

We apply the following six integration schemes to the four-compartment plankton system (Eqs. (14a)–(14d)): fixed time-step Heun Runge–Kutta, BBKS2, mBBKS2, eBBKS2, samBBKS2 and adaptive time-step Heun. We assess their performance by three measures of accuracy:

$$E_1(n\Delta t) = \frac{1}{I}\sum_{i=1}^{I}\sqrt{\frac{(c_i(n\Delta t)-c_i^n)^2}{(c_i(n\Delta t))^2}} = \frac{1}{I}\sum_{i=1}^{I}\left|\frac{c_i(n\Delta t)-c_i^n}{c_i(n\Delta t)}\right|, \tag{16a}$$

$$E_2 = \frac{1}{N}\sum_{n=1}^{N} E_1(n\Delta t), \tag{16b}$$

$$E_3 = \frac{1}{I}\sum_{i=1}^{I}\sqrt{\frac{\sum_{n=1}^{N}(c_i(n\Delta t)-c_i^n)^2}{\sum_{n=1}^{N}(c_i(n\Delta t))^2}}. \tag{16c}$$

Here, $c_i(n\Delta t)$ denotes a high-quality (i.e. accurate) solution for state-variable I at time $n\Delta t$. Following BBKS, the accurate values were generated using a fourth-order Runge–Kutta scheme with $\Delta t = 0.01$. E_1 is the Manhattan norm of relative errors and provides a measure of the instantaneous relative error between the (possibly) low-quality solution, and the high quality one. E_2 is an approximate average error over all time-steps. E_3 is an alternative measure of the time-averaged relative error. It is the measure used by BBKS, but is less sensitive to phase-errors than E_2.

For the Robertson system, we compare the performance of the BBKS2, mBBKS2, eBBKS2 fixed-time step schemes at time-steps of 0.0005 and 0.001. We also examined the performance of the samBBKS2 and adaptive time-step Heun schemes (using a nominal, i.e. maximum) time-step of 0.001. In addition to assessing accuracy, we measured the duration of each simulation to assess each method's speed.

For both the plankton-system, and the Robertson system, the accuracy criterion for c_i^{n+1} in the adaptive-Heun scheme was set proportional to c_i^n (and independent of $f(t^n, \vec{c}^n)$—see 16.2.8 and 16.2.10 of Press et al. [7]). Furthermore, should any c_i^{n+1} prove to be less than or equal to zero, the projection was repeated with a smaller time-step (chosen to be the root of the linear interpolant for the offending state-variable across the offending time-step). If required, time-step reduction continued until positive solutions were generated.

For the samBBKS2 scheme, the internal-time-step for each iteration was chosen such that the gradient modifier term for the predictor step of the mBBKS scheme would exceed 0.9999. For both the Robertson system and the plankton system, we applied the eBBKS scheme with $\beta = 0.9999$, $\beta = 0.99$ and $\beta = 0.90$.

6. Results

6.1. Simple non-linear system

Figs. 1(a) & (b) compare the discrepancies between the solutions to Eqs. (14a)–(14d) stemming from the BBKS2, mBBKS2 and the explicit second-order Heun-type Runge–Kutta method relative to a high-accuracy solution stemming from a fourth-order Runge–Kutta scheme.

As one would expect, the performance of all of the second-order schemes declines as the time-step increases. In the case of the fixed-time-step Heun method, this performance loss increases dramatically once the time-step becomes so large that the method generates unstable results (instability arises at time-steps greater than approximately 0.64; negative projected state-variables (in either $c_i^{(1)}$ or c_i^{n+1}) first appear at a time-step of approximately 0.32). eBBKS2 and mBBKS2 clearly outperform BBKS2 at all time-steps. At fine time-steps, and depending upon whether error-measure E_2 or E_3 is used, mBBKS2 is either marginally superior or marginally inferior to the explicit fixed time-step Heun method (and eBBKS2—which operates as the Heun method at time-steps less than approximately 0.32). At time-steps larger than 0.64, the BBKS2 and mBBKS2 methods outperform the fixed time-step Heun method, but eBBKS2 (with $\beta = 0.9999$) performs still better. eBBKS2 with $\beta = 0.99$ is also superior to mBBKS2 at large time-steps (not shown), but with $\beta = 0.9$, eBBKS2 proves inferior (not shown).

Fig. 2 confirms that, as anticipated, the errors E_2 and E_3 arise because the gBBKS and eBBKS methods have a tendency to slow the time-averaged reaction rates excessively at large time-steps—thereby introducing phase-errors.

Fig. 3 compares the discrepancies between the solutions to the simple plankton system (Eqs. (14a)–(14d) stemming from the explicit, fixed time-step Heun method, and the two adaptive time-step methods: samBBKS2 (implicit) and adaptive-Heun (explicit) relative to those stemming from a fourth-order Runge–Kutta scheme with a time-step of 0.01.

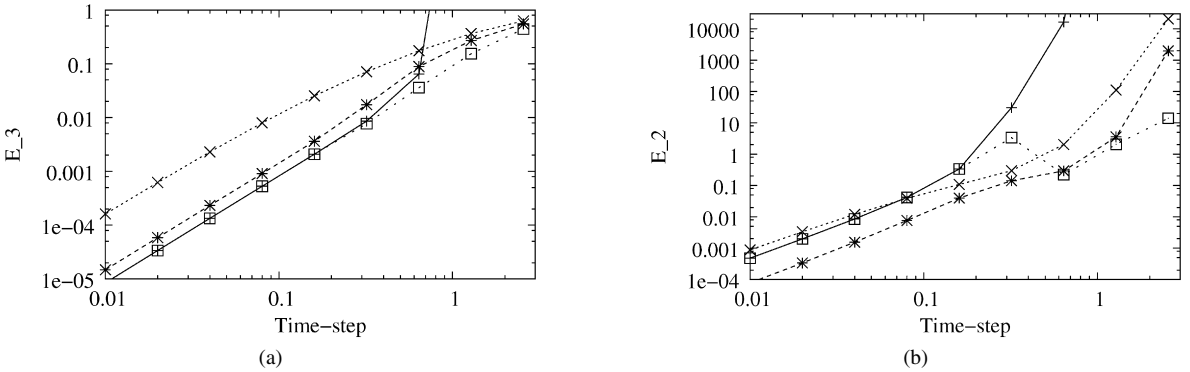

Fig. 1. Relative discrepancy ((a) measured by E_3, Eq. (16c); (b) measured by E_2, Eq. (16b)) versus time-step for standard Heun-method (+), the BBKS2 method (×), the mBBKS2 method (star) and the eBBKS2-method ($\beta = 0.9999$, square) relative to solutions stemming from a fourth-order Runge–Kutta scheme with $\Delta t = 0.01$. The catastrophic loss of accuracy exhibited by the Heun method arises when it begins to generate unstable solutions.

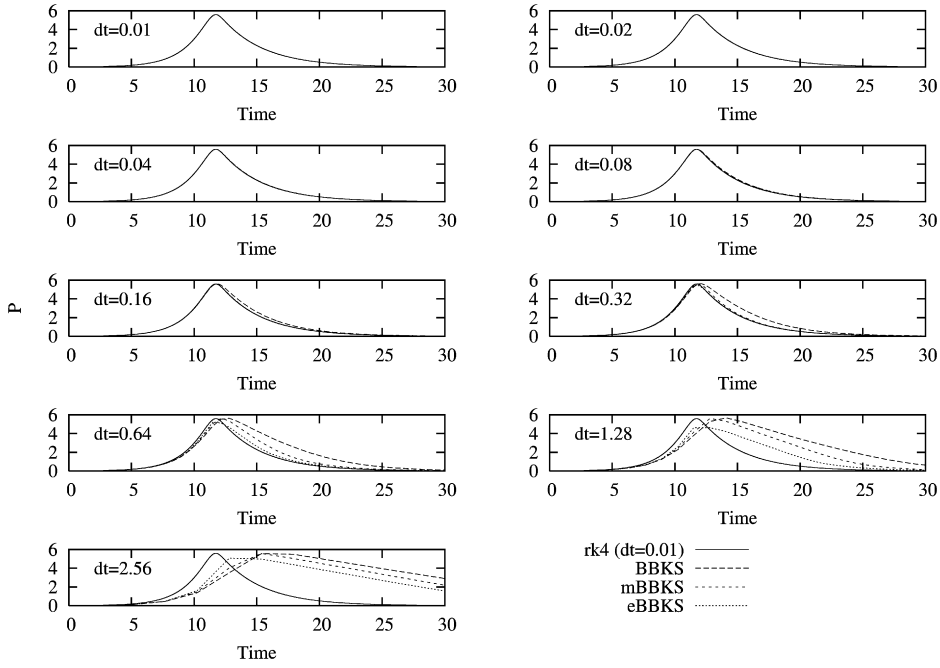

Fig. 2. Simulated abundance of state-variable P (Eq. (14c)) calculated using different integration schemes (line-styles) and (for methods other than fourth-order Runge–Kutta) time-steps (panels). For the eBBKS-scheme, $\beta = 0.9999$. The time-step is indicated within each panel.

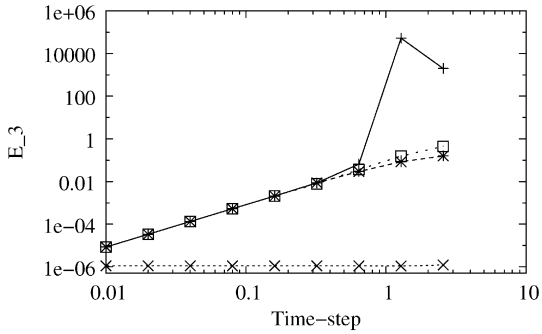

Fig. 3. Relative discrepancy (E_3, Eq. (16c)) versus time-step for standard Heun-method (+), the samBBKS2 method (×), adaptive time-stepping Heun method (star) and the eBBKS2-method ($\beta = 0.9999$, square) relative to solutions stemming from a fourth-order Runge–Kutta scheme with $\Delta t = 0.01$. The catastrophic loss of accuracy exhibited by the Heun method arises when it begins to generate unstable solutions. For the adaptive time-stepping Heun method, the error tolerance was chosen such that accuracy would be similar to the mBBKS method (see Fig. 1). Specifically, the tolerable error for state-variable C_i^{n+1} was set to $1.25C_i^{n+1}$. This is unusually lax, however the time-step was reduced and the projection repeated whenever C_i^{n+1} proved to be less than zero. For the samBBKS2 method, the size of each internal time-step was chosen such that the gradient modifier for the predictor-step of the scheme exceeded 0.9999.

In the case of the adaptive-Heun method, the error-scaling constant was chosen such that the resultant errors were of similar magnitude to those stemming from the mBBKS method (specifically, it was set to 1.25; furthermore, should any state-variable prove negative at the end of the time-step, the projection was repeated with a smaller time-step). As one would hope, the two adaptive methods perform as well as, or better than the fixed time-step Heun method. For comparison, the error ratios arising from the adaptive-Heun method applied with a much more stringent error-scaling-constant (10^{-5}) are of order 10^{-6} (i.e. an order of magnitude smaller than those achieved with the comparatively large acceptable error ratio used in Fig. 3, and similar to those achieved by samBBKS2).

Table 2
Relative runtime of the integration schemes applied to the Robertson system run to time = 2000

Scheme	Relative run-time	Number of right-hand-side evaluations relative to BBKS2
BBKS2	1.05	1.0
mBBKS2	1.03	1.0
eBBKS2	0.98	1.0
samBBKS	1.33	2.24
Adaptive Heun	1	2.51
Fixed time-step Heun	1.21	4.41

The nominal time-step was 0.0005. For the samBBKS scheme, internal time-steps were chosen such that the gradient-modifier term for the predictor-step would exceed 0.9999. For the eBBKS-scheme, $\beta = 0.9999$. For the adaptive Heun scheme, the error tolerance was $0.01C_i^n$. For both the adaptive-Heun, and fixed time-step Heun methods, the time-step was reduced and the projection repeated if C_i^{n+1} proved negative.

6.2. Robertson system

The Robertson system presents a more demanding test of the numerical schemes. This system takes in excess of 10^6 time-units to come to equilibrium, but it was not practical to make simulations of that duration because the BBKS, mBBKS and eBBKS schemes all came to yield highly inaccurate results at relatively small time-steps. At $\Delta t = 0.0005$, both the BBKS2 scheme, and the mBBKS2 scheme yield moderately accurate reproductions of the Robertson system dynamics (as far as time = 2000). Those stemming from the mBBKS2 scheme are, perhaps, superior (Fig. 4(a)). Using $\Delta t = 0.001$, both schemes yield results which soon diverge dramatically from the true solution (Fig. 4(b)). Despite the ultimate failure of both BBKS2 and mBBKS2 at this time-step, inspection of Fig. 4(b) leads one to conclude that the latter method performs better than the former until each begins to diverge catastrophically from the accurate solution. In contrast to the pattern for the simple plankton system, mBBKS2 yields more accurate solutions to the Robertson equations than eBBKS2 does ($\Delta t = 0.0005$; compare mBBKS forecasts in Fig. 4 with those for eBBKS in Fig. 5). With a time-step of 0.001, the eBBKS2-scheme yields dynamics (not shown) similar to those of BBKS2 and mBBKS2 schemes at this time-step (i.e. system evolution comes to a near standstill, giving the impression of a 'plateau').

Unlike their fixed time-step counterparts, the adaptive schemes (samBBKS2, and adaptive Heun-coefficient explicit Runge–Kutta) both yield qualitatively correct long-term results when run with a nominal time-step of 0.001 (albeit that the realized time-steps must have been smaller than this on at least some occasions). Results for the samBBKS2 scheme are illustrated in Fig. 5. Those stemming from the adaptive Heun-scheme (with an error tolerance of $0.01c_i^n$, and reversion with time-step reduction should any c_i^{n+1} prove negative) are a little more accurate than those of samBBKS2 (not shown).

Table 2 reveals the relative computational demands of the various schemes. eBBKS, adaptive Heun, BBKS and mBBKS are all of similar speeds. The fixed-time-step Heun and samBBKS2 schemes are markedly slower. In the former case, the poor performance is because the prescribed time-step (0.0005) did not always permit positive solutions. Thus, the scheme frequently had to revert to the beginning of a time-step and repeat the projection using a smaller time-step (i.e. the scheme was not genuinely fixed time-step, but no *a priori* attempt was made to choose each forthcoming time-step such that it would be likely to yield positive results). The samBBKS2 scheme is so much slower because, given the criterion on the minimum acceptable value for the gradient-modifier associated with the predictor-step, it was forced to take numerous internal time-steps to traverse an external step. Interestingly, despite being much slower than the adaptive Heun method, it was also less accurate.

7. Discussion

7.1. Fixed time-step interpretation

BBKS aimed to develop an unconditionally positive, mass-conserving, higher-order fixed time-step scheme. They did so by developing a scheme which automatically, and smoothly slows all of the reaction kinetics—particularly when one or more state-variables is in danger of becoming negative. The mBBKS (more generally, gBBKS) schemes

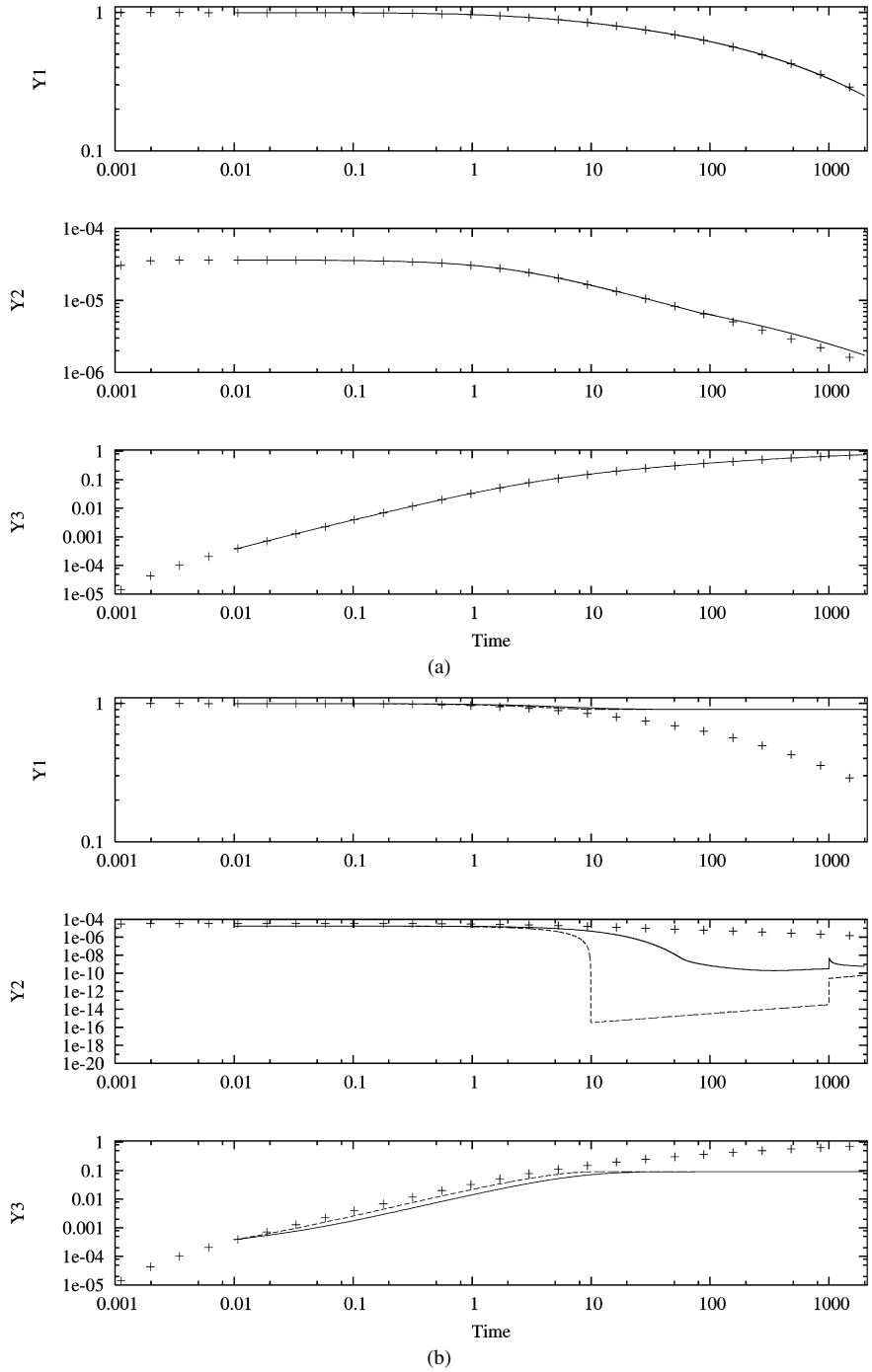

Fig. 4. (a) Numerical solutions of the equations defining the Robertson system ($\Delta t = 0.0005$, but output interval $= 0.01$). Solid line: BBKS2 scheme; dashed line: mBBKS2 scheme (not evident because the results overlap those of the BBKS2 scheme); crosses: solution generated by the Matlab® library-file hb1dae.m. (b) Numerical solutions of the equations defining the Robertson system ($\Delta t = 0.001$ with output at intervals of 0.01). Solid line: BBKS2 scheme; dashed line: mBBKS2 scheme; crosses: solution generated by the Matlab® library-file hb1dae.m.

work in the same manner, but, because it is scale-independent, performs better at all time-steps when more than one state-variable is in decline. The improvement is most marked at small time-steps. Whilst scale-independence is clearly desirable in a spatially explicit system (such as the reaction/diffusion one referred to in the introduction), it transpires

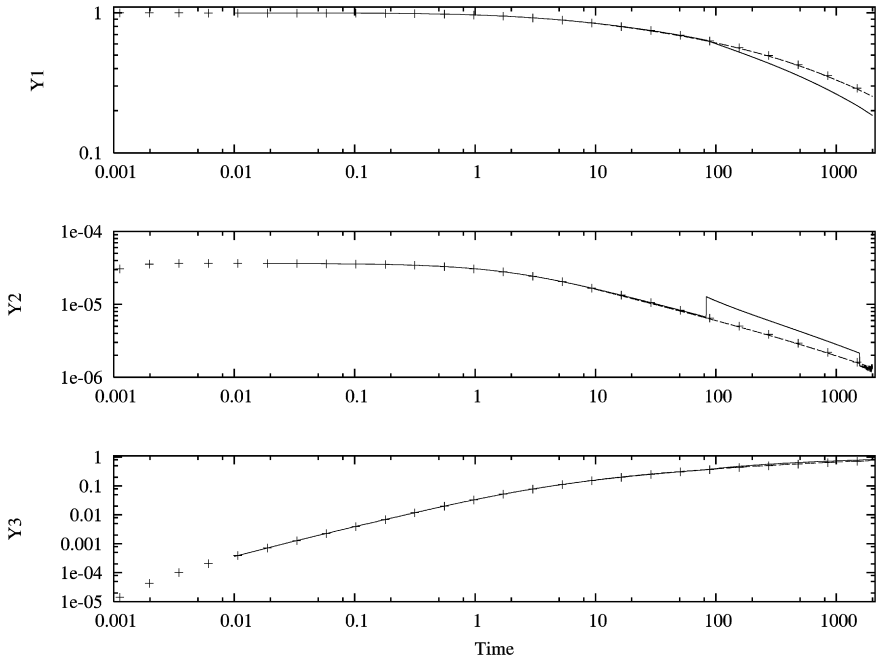

Fig. 5. Numerical solutions of the equations defining the Robertson system. Solid line: eBBKS2 scheme ($\Delta t = 0.0005, k = 0.9999$; dashed line: samBBKS2 scheme (external time-step = 0.001, but internal time-steps chosen such that gradient modifier exceeds 0.9999); crosses: solution generated by the Matlab® library-file hb1dae.m.

that it is also beneficial in a spatially homogeneous system such as the simple plankton system and the Robertson system that we have examined in detail. This reason is that by requiring the gradient modifier to be scale-independent, we have caused the resultant modifier term to diverge from one by a lesser margin. Thus, the resultant scheme remains more nearly faithful to a Taylor-expansion (with the result that the system's reactions experience less damping).

Theorem 1. *For any system having more than one declining state-variable the departures induced by mBBKS are always smaller than those introduced by BBKS* (*i.e.* $m(\vec{f}, \vec{c}^n)$ *is closer to* 1.0).

Proof. The modifier $m(\vec{f}, \vec{c}^n)$ we seek for the BBKS-scheme is p_r, say, where p_r is the root of the polynomial $g(p_r) = 0$, where $g(\tilde{p}) = \prod_{j\in J^n}(1 + a_j\tilde{p}) - \tilde{p}$. In the BBKS scheme $\tilde{p} = p$, where p is defined by Eq. (5c), viz $p = \prod_{j\in J^n} \frac{c_j^{n+1}}{c_j^n} > 0$. For mBBKS we seek z_r, such that $h(z_r) = 0$, where $h(z) = \prod_{j\in J^n}(1 + a_j z) - z^J$, where the modifier z now takes the form $z = \sqrt[J]{p}$. As BBKS showed, $g(0) = 1, g(p_{\max}) < 0$, and $g(\tilde{p})$ is a continuously decreasing function between these limits, hence the existence of exactly one real root p_r in this range. The behavior of $h(z)$ is similar. Considering both polynomials in terms of the same dependent variable x, say, we have $h(x) - g(x) = x - x^J$. Hence we can locate z_r by considering the effect of adding $x - x^J$ to $g(x)$. In the range $0 < x < 1$ the function $x - x^J$ is always greater than zero. From the form of $g(x)$ detailed above we know that it is positive/negative below/above the root p_r. Hence the root z_r can only lie above p_r, and so the mBBKS departures are always smaller than those from BBKS. Thus, it is inevitable that mBBKS will outperform BBKS at sufficiently small nominal time-steps.

Our simulations clearly demonstrate that, despite maintaining positivity, the fixed time-step methods (BBKS, mBBKS, eBBKS) fail to yield quantitatively accurate solutions when the time-step becomes too large. Indeed, the methods fail to reproduce even the qualitative dynamics of the Robertson system at large time-steps—though their failure is less catastrophic for the plankton system. Positivity is an important accuracy standard, but a scheme operating by this requirement alone cannot guarantee to generate accurate solutions. In part this is because it takes no steps to ensure accuracy when all declining state-variables are far from zero, however even when state-variables are in rapid relative decline, seeking to maintain positivity cannot be guaranteed to equate to seeking to maintain results which are accurate (see below).

As anticipated, eBBKS proved to be a little faster than mBBKS, but it was not invariably more accurate. The two schemes are identical in that they rely upon applying a gradient-reduction scalar to the calculated rates of change. They differ only in the manner in which the gradient reduction term is calculated. In the eBBKS-scheme it is determined by the magnitude of only the fastest state-variable-specific-relative-rate-of-decline and by the value of β (which might be hard-coded, or user-specified). The former determines the maximum acceptable time-step (maximum acceptable gradient-modifier for this state-variable), the latter determines how much additional rate reduction (i.e. damping) is applied. In the gBBKS-scheme, all of the declining state-variables contribute to determining the realized size of the gradient modifier. The relative contribution of each (and, hence, the degree of additional damping) is determined by the value of q (likely $= rJ$, see Eq. (12) and ensuing text). Given that, in the eBBKS-scheme, the instantaneous value of the gradient modifier is determined by only one state-variable, whilst it is (loosely-speaking) determined by the average of several state-variables in gBBKS-schemes, one might anticipate that the gradient modifier would prove to be a little more temporally stable in gBBKS-schemes. We do not know whether or not this is a desirable property for an integration scheme.

A key question that applies equally to the gBBKS and eBBKS-schemes is: 'how much additional damping (in excess of that required to preclude negativity) is required to yield a quantitatively accurate projection?'. This is tantamount to admitting that the gradient-modifier should be viewed not solely as a means of maintaining positivity, but also as a means of approximating the rate-contribution made by higher-order terms (that are neglected in an explicit Runge–Kutta scheme's approximation to the Taylor expansion). As such, there must be an optimum value for the gradient-modifier term. Identifying this optimum value is equivalent to identifying the optimum value for q (or r) in the gBBKS scheme, or β in the eBBKS scheme. We suspect that there is no universal optimum. Rather, the instantaneous optimum will depend upon the details of the system (functional form and coefficients of the differential equations, instantaneous state-variable values) and the time-step.

Identifying a means by which an approximation to the instantaneous optimum might is beyond the scope of this paper, but we offer the following qualitative suggestion for the second-order gBBKS and eBBKs schemes: when the gradient-modifier term associated with the predictor step is sufficiently close to 1.0 and the instantaneous rates of change ($\vec{f}^{(1)} = f(t^{n+1}, \vec{c}^{(1)})$) estimated at $t = t^{n+1}$ with the preliminary estimates of the state-variables at this time (i.e. $\vec{c}^{(1)}$) are sufficiently similar to those estimated at $t = t^n$ with $\vec{c}^n$, the systems is in a near-linear region and is unlikely to be in danger of yielding a negative state-variable, and the second-order Taylor polynomial expansion is likely to be accurate. Thus, one might choose to estimate the gradient-modifier (m_2) that will be applied during the corrector step such that it will yield little more than the minimum-required damping (i.e. eBBKS: choose, β only marginally less than 1.0; scale-independent gBBKS, choose r much larger than 1). This ensures that the corrector projection remains faithful to the second-order projection. When the gradient-modifier term associated with the predictor is not sufficiently close to 1.0, or the two vectors of rates-of-change are not similar, there is a lesser likelihood that the second-order Taylor expansion will be accurate, and one might choose to estimate the gradient-modifier for the corrector-step such that it will induce more than the damping that is required solely to maintain positivity (eBBKS: choose β rather less than 1.0; scale-independent gBBKS: choose r little larger than 1.0).

At very large time-steps, $m(\vec{f}, \vec{c}^n)$ converges towards zero regardless of the value of r (or, in the eBBKS-scheme, β). This explains the converging time-averaged performance of BBKS and mBBKS as the time-step increases. All gBBKS-schemes can be expected to behave in this manner. Nonetheless, because $m(\vec{f}, \vec{c}^n)$ will diverge from 1 less rapidly when r is large, large-r gBBKS schemes will be more prone to yielding a projected state-variable value of zero (due to round-off error) than small-r ones. Furthermore, even when positivity is not violated, it is likely that large-r schemes will be more prone to inducing spurious oscillations than small-r schemes (because the former induce less damping; the same problem will afflict the eBBKS-scheme if β is set too close to 1.0).

Recognizing that gBBKS- and eBBKS-schemes operate by slowing local-scale reaction kinetics, one must question how best to simulate systems in which the coupling amongst state-variables is imperfect. Firstly, let us re-consider the earlier system of reactor-vessels coupled by diffusion. This time, let us assume that the initial conditions differ among reactors—such that if the system of ODEs were solved using a traditional, explicit scheme (such as RK2 with a coarse time-step), only one of the vessels would be in imminent danger of suffering a negative reagent concentration. If the entire system of ODEs were solved simultaneously using a gBBKS-method, negativity in this one vessel would be eliminated—but at the cost of unnecessarily reducing the rates of change in the remaining vessels. Alternatively, one might choose to apply the gBBKS-method to each vessel in turn (and solving for the dispersion contributions separately, as envisaged by BBKS). This would eliminate unnecessary slowing of reactions in other vessels, but incur

the cost of possibly introducing phase-errors between vessels (the computational burden would also increase because more implicit polynomials would need to be solved). Depending upon the nature of the reaction kinetics and the strength of the dispersive coupling, the 'improved' short-term vessel-specific performance may induce longer-term vessel-specific dynamics which differ from the true dynamics by more than phase (i.e. the shape of the attractor may be modified).

As a second example, we note that imperfect coupling can occur even at the local-scale. In contrast to the assumption made in the simple nutrient/phytoplankton/detritus model described in Eqs. (14a)–(14d), phytoplankton do not maintain absolutely fixed stoichiometries. The dominant constituent of phytoplankton biomass is usually carbon (in some diatoms the silicon frustule can dominate), but even within a single cell, the elemental composition (e.g. N : C, P : C and Si : C ratios) can independently vary markedly (albeit within bounds which are empirically moderately well known) over time under the influence of changing environmental conditions (light intensity, water temperature, ambient nutrient concentrations). Consider a case where a population of nutrient-limited phytoplankton is introduced to water rich in N and P but not Si—as might occur near a point-source ocean outfall. At their entry into this water, the phytoplankton will likely have low N : C, P : C ratios and Si : C ratios. In reality, their N : C ratio and P : C ratio will climb rapidly, but under a gBBKS/eBBKS scheme, this rapid increase will be suppressed if the system's state is such that the ambient concentration of dissolved inorganic silicon is in decline.

Some models of phytoplankton dynamics even distinguish differing intra-cellular tissue-types (for example, 'composite cellular structural material' and 'elemental reserves' from which structural materials are synthesized [5]). A phytoplankter moving vertically (under the influence of turbulence, or through swimming/buoyancy-regulation behaviors) between nutrient-poor surface waters and nutrient-rich, but dark deeper waters, is likely to suffer alternating periods in which the internal nutrient and carbon reserve pools become depleted. Phase errors in the replenishment of either will modify the emergent rate of synthesis of structural material, and hence the demographic growth rate.

At first sight, it might be thought that the problems arising from imperfect elemental coupling could be alleviated by considering local-scale state-variable dynamics to be independent of one another over the course of a single time-step (i.e., in a model posed in terms of C, N, P and Si, by calculating independent values of $m(\vec{f}, \vec{c}^n)$ for C, N, P and Si). With some caveats (see below), this approach will work when the state-variables are elements. Unfortunately, as shown by BBKS, if the state-variables are compounds (e.g. protein), it cannot be invoked without violating mass-conservation. The caveats that we mentioned are as follows. Firstly, the approach will increase the computational burden. Secondly, the realised N : C, P : C and Si : C ratios may stray slightly beyond the respective user-supplied bounds. This cost is likely to be acceptable because: (a) these bounds are known only empirically, and to coarser precision than likely magnitude of stray, and (b) the homeostatic mechanisms built into models that permit variable stoichiometries should counter this stray during subsequent time-step(s).

7.2. Adaptive-time-step interpretation

We have established that though originally posed as fixed-time-step scheme by BBKS, gBBKS-type schemes can be reformulated as novel, adaptive time-step schemes. Unlike their fixed-time-step progenitors, these adaptive schemes do not introduce spurious time-lags. Thus, there are no difficulties when simulating systems that exhibit imperfect coupling.

Adaptive time-step schemes usually operate by one, or both, of two manners:

(a) A time-step is chosen (perhaps on the basis of the error estimate from the previous time-step), and two differing projections across this time-step are made. The differences between corresponding state-variables at the end of the times-step are compared with a predefined tolerance [7]. If the tolerance is exceeded, the projection is repeated over a shorter time-step.
(b) In cases where there are known state-variable thresholds that cannot be crossed (or at which some discrete event takes place), the value of each of these state-variables is compared with the associated prescribed thresholds after each time-step. The time-step is reduced and the projection is repeated whenever the threshold has been crossed by an unacceptably large amount.

Our adaptive time-step Heun integration scheme used both (a) and (b). The samBBKS2 scheme can be viewed as working in manner that is akin to (a). The size of the difference between $m(\vec{f}, \vec{c})$ and 1.0 provides an indication of the

extent to which a simple mBBKS projection would differ from a (equivalent order) Taylor-polynomial projection. If this difference is deemed too large, the internal time-step is reduced.

In our trials, samBBKS2 proved to be substantially less efficient than a more traditional adaptive-time-step scheme—despite making fewer evaluations of the ODE right-hand sides (Table 2). It is difficult to know whether the samBBKS2 scheme will invariably be more-, or less computationally demanding than a traditional adaptive, explicit Runge–Kutta scheme. The samBBKS2 scheme may require repeated calls to an iterative root-finding algorithm, but a traditional adaptive scheme requires calculation of error tolerances, comparison of these with the inferred errors of the projection, and projection repetitions whenever the error tolerances have been exceeded. The relative performance of samBBKS2 and traditional adaptive schemes will almost certainly depend upon: (a) the cost of evaluating the ODE right-hand sides relative to that of solving the implicit polynomial, and (b) the frequency with which the preliminary time-step chosen by a traditional adaptive scheme subsequently proves to be sub-optimal (whether too large or unnecessarily small). □

8. Conclusion

The original intent of BBKS was to develop a guaranteed positive, computationally cheap, high-order fixed time-step scheme applicable to non-linear systems. They achieved this—at the cost of introducing phase-errors into the simulation results when the time-step is too large. We have shown that these can be dramatically reduced (but not eliminated) by changing the formulation of the modifier term of their scheme. They can be eliminated by recognizing that the gradient-modifier term can be treated as a time-step modifier term, and reformulating the schemes as adaptive-time-step ones.

Our fixed-time-step improvement to the original BBKS-scheme is bought at negligible cost in terms of difficulty of implementation or computational burden, and can be generalized to yield the gBBKS scheme. gBBKS methods operate by artificially slowing all reaction-rates when even a single state-variable is in imminent danger of becoming negative. Recognizing that positivity is the sole accuracy criterion by which gBBKS-schemes select the gradient modifier term, led to the development of the eBBKS scheme. This is slightly faster than the gBBKS scheme because it does not require that an implicit polynomial be solved. The key to further improvement in both gBBKS and eBBKS will be in identifying the optimal (in the sense of yielding accurate numerical integrations) values for the parameters in the two schemes (respectively q or r and β). An important associated question is whether the slightly different temporal patterns of damping implied by gBBKS and eBBKS schemes imply that one or other is more likely to be superior when applied to an arbitrary problem.

Whilst a superior, truly fixed-time-step scheme may yet be found, a more profitable approach may be to ensure that the time-step-control algorithms for the numerical simulation of coupled physical/biochemical systems (e.g. coupled hydrodynamic/nutrient/phytoplankton models) take account of constraints associated with developing adequate solutions to local-scale reaction processes, as well as those associated with developing adequate solutions to advection/dispersion problems (as is already done in some plankton models [1]). In the meantime, for systems which are not overly stiff, the gBBKS or eBBKS methods may prove useful when phase-errors are of little concern and it is not possible to implement adaptive-time-stepping schemes.

Acknowledgements

This work was stimulated by work undertaken on behalf of Environment Waikato. Niall Broekhuizen and Graham Rickard were funded by the New Zealand Foundation for Research, Science & Technology (contract C01X0501). The work of Jorn Bruggeman was supported by the Netherlands Organization for Scientific Research (NWO) through grant number 635.100.009. The work of Andreas Meister was supported by the Deutsche Forschungsgemeinschaft within the project C3 of the SFB/Transregio 30. Our colleague Graham McBride provided helpful responses to earlier versions of this manuscript. Our co-authors Dr. Jorn Bruggeman and Prof. A. Meister were the referees of the original version of this manuscript. Dr. Bruggeman recognized that the BBKS and mBBKS schemes could be generalized to yield the gBBKS scheme. Prof. Meister suggested the eBBKS-scheme as a possible alternative to mBBKS.

Appendix A. Pseudo-code illustrating how the second-order gBBKS, eBBKS and their adaptive time-step variants might be implemented

```
/*=========================================================================*/
/* calculate the value for the gradient modifier.                          */
/* WARNING, WARNING, WARNING:                                              */
/*          For schemes other than eBBKS:                                  */
/*          This gradient-modifier is slightly different from              */
/*          the one referred to in the text in that it is the              */
/*          product of a root-derived modifier term and the                */
/*          inverse-change-ratio term (see Eq. 3c).                        */
/* method:  flag specifying what sort of integration method                */
/* sv0:     vector of state-variable values at start of present            */
/*          time-step.                                                     */
/* cnk_by_c1k: scalar inverse change-ratio (=1.0 for predictor-            */
/*             step, and based on quotient sv/sv^(1) for                   */
/*             corrector step)                                             */
/* r:       scale-factor of gBBKS scheme                                   */
/* grad:    vector of rates of change: corresponding to either             */
/*    a)    g1: rates of change at start of time-step                      */
/*    b)    ga: prelim estimate of time-step averaged rate of              */
/*              change: 0.5*(g1+g2)                                        */
/* internal_dt: size of the proposed time-step                             */
getGradientModifier(method,r,sv0,grad,cnk_by_c1k,internal_dt){
  if(eBBKS==method){
    beta=1-eps; /* where eps is a user-specified,               */
                /* or hard-coded, small, positive number        */
    /* now, find the largest acceptable time-step */
    acceptable_dt=internal_dt;
    for(i=0;i<n_svar;i++){
      if(0.0>g[i]){
        largest_dt= -sv[i]/grad[i];
        if(largest_dt<acceptable_dt){
          acceptable_dt=largest_dt;
        }/* if largest_dt */
      }/* if 0>g[i] */
    }/* for i */
    if(acceptable_dt<=internal_dt/beta){
      return(beta*acceptable_dt/internal_dt);
    }
    else{
      return(1);
    }
 }
 else{
    solve for p using implicit eqn g(p,r,sv0,grad, cnk_by_c1k)
    derive gradient modifier from p
    /* return a realized modifier term that takes account    */
    /* of the inverse-change ratio (==1 for predictor step)  */
    return(gradient modifier*cnk_by_c1k)
 }
```

```
}/* end getGradientModifier() */
/*=============================================================================*/

/*=============================================================================*/
/* calculate the realised size of the time-step.      */
/* Unless the integration-method is amBBKS, this      */
/* simply returns the proposed size of the time-step  */
/* For amBBKS, it recognises that the BBKS-type       */
/* methods can be viewed as 'slowing time'.           */
getRealised_dt(internal_dt,gradient_modifier,method){
   if(amBBKS==method){
     return(gradient_modifier*internal_dt)
   }
   else{
     return(internal_dt);
   }
}/* getRealised_dt() */
/*=============================================================================*/

/*=============================================================================*/
/* Now the code to implement the gBBKS/eBBKS scheme   */
/* (including one internal cycle of the adaptive      */
/* variants).                                         */
/* State variables are projected over a single        */
/* (perhaps internal) time-step.                      */
/* with r=1/J, this corresponds to BBKS2, with r=1,   */
/* this corresponds to mBBKS                          */
/* sv: vector of stave vars at t0                     */
/* t0: time at start of proposed time-step            */
/* external_dt: size of proposed time-step            */
/* r:  scaling term of qBBKS scheme.                  */
/* min_m1:      minimum acceptable size of the        */
/*              gradient-modifier associated with     */
/*              the predictor-step. Set to 0 for      */
/*              BBKS,mBBKS (gBBKS), or 0<min_m1<1     */
/*              for samBBKS, (sagBBKS).               */
/* cancel_phase_lag: TRUE -> amBBKS-like scheme       */
/*                   FALSE -> mBBKS-like or samBBKS   */
/* method: flag indicating whether we are using       */
/*       (i)    some variant of gBBKS (incl.          */
/*              BBKS, mBBKS, amBBKS, samBBKS, and     */
/*              variants upon this with r>1)          */
/*       (ii)   eBBKS                                 */
int ok=gBBKS(sv, t0, external_dt, r, min_m1, cancel_phase_lag, method){
internal_dt=external_dt;
internal_time=t0;

/* some parameter checks. Would probably put these somewhere                */
/* earlier in the simulation-code so that they are checked                  */
/* only once rather than at every time-step.                                */

/* disallow combination of amBBKS-like and samBBKS-like schemes             */
```

```
if(TRUE==cancel_phase_lag && 0.0<min_m1){
  printf("ERROR: you cannot invoke gBBKS() with parameters");
  printf(" cancel_phase_lag=TRUE, and min_m1 >0");
  printf("implying simulataneous use of amBBKS-like and samBBKS-like methods");
  printf("This combination is redundant and I have chosen to forbid it.")
  /* note potential memory leak */
  exit(EXIT_FAILURE);
}
if(0.0>min_m1){
   printf("ERROR: gBBKS() invoked with 0>min_m1. Require 0<=min_m1<1);
   /* note potential memory leak */
   exit(EXIT_FAILURE);
}
else{
   if(1<=min_m1){
     printf("ERROR: gBBKS() invoked with 1<=min_m1. Require 0<=min_m1<1);
     /* note potential memory leak */
     exit(EXIT_FAILURE);
   }
}
if(0.0>r){
   printf("ERROR: gBBKS() invoked with 0.0>r. Require 0<=r");
   /* note potential memory leak */
   exit(EXIT_FAILURE);
}
else{
   if(1>r){
     printf("WARNING: gBBKS() called with 0<=r<1, implying that the ");
   printf(" integration scheme will yield a gradient-modifer which ");
   printf(" is concave w.r.t. change-ratio. Better to choose r>=1.");
   }
}
/* start the predictor-step                                          */
/* calling getGradientModifier() with argument cnk_by_c1k = 1.0      */
/* for predictor-stage.                                              */
calculate g1=temporal gradients at t=t0 with state-vars = sv
calculate m1=getGradientModifier(method,r,sv,g1,1.0,internal_dt);
/* may need to shrink internal_dt if method is samBBKS-like */
while(m1<min_m1)
   internal_dt=0.5*internal_dt;
   calculate m1=getGradientModifier(method,r,sv,g1,1.0,internal_dt)
}
project sv1 = sv+internal_dt*m1*g1
project t1 = t0+getRealised_dt(internal_dt,m1,method)
if(TRUE==cancel_phase_lag){
   /* recognize that time has slowed and dt is smaller               */
   /* than anticipated                                               */
   internal_dt=t1-t0;
}
/* now calculate gradients at end of time-step */
calculate g2=temporal gradients at t=t1 using sv1
/* calculate the inverse change ratio (cnk_by_c1k)                   */
```

```
/* for declining state vars                                          */
cnk_by_c1k=1.0;
for(i=0;i<n_svar;i++){
   if(0.0>(g1+g2)){
     cnk_by_c1k *= sv1[i]/sv[i];
   }
}
ok, now the gradient modifier for the ultimate, corrector projection        */
/* Note that this term will include the effect of the inverse (prelim.)     */
/* change-ratio cnk_by_c1k                                                  */
Calculate m2 = getGradientModifier(method,r,sv,0.5(g1+g2),
                                   cnk_by_c1k,internal_dt)

realised_dt = getRealised_dt(internal_dt,m2,method)

/* project state vars to end of time-step. Apply appropriate weights to     */
/* the two gradient estimates if this is an amBBKS-like scheme.             */
if(TRUE==cancel_phase_lag){
   wt_g1= 1.0-m2/2.0;
   wt_g2= 1.0-m1;
else{
   wt_g1=0.5;
   wt_g2=0.5;
}
/* make the 'corrector' projection across realized time-step                */
/* note that if this is an amBBKS-like scheme, internal_dt has              */
/* already been shrunk (as a result of the first projectn). It              */
/* will shrink again. If this is not an amBBKS-like scheme                  */
/* internal_dt retains the value that was initially calculated              */
/* before making the predictor-projectn. It will continue to                */
/* retain this value. Remember that m2 already includes the                 */
/* influence of the inverse change-ratio term                               */
project sv_next=sv+internal_dt*m2*(wt_g1*g1+wt_g2*g2);
/* also evolve the time (if this is amBBKS-like the net rate                */
/* of evolution will be in range 0<rate<1. Otherwise, it                    */
/* will be 1.0                                                              */
project t_next=t0+realised_dt
return t_next, sv_next, realised_dt
}/* end gBBKS() */
/*==========================================================================*/
```

Appendix B. An alternative adaptive time-step algorithm (amBBKS)

In the main-body of the text we present the samBBKS-scheme. That scheme operates by ensuring that any accruing phase error is kept within acceptable bounds. Here, we describe an alternative, but more complex scheme (dubbed amBBKS) that eliminates the phase-error. This is achieved by applying the gradient modifier term to the rate at which simulated time evolves, as well as to the rate at which the state-variables evolve. It achieves this by applying the gradient modifier $m(\vec{f}^n, \vec{c}^n)$ to the evolution of time, as well as the evolution of the state-variables. In order to maintain temporal consistency of higher order schemes, several additional steps must be taken:

(1) this reset must be made prior to every individual evaluation of the ODEs in order to ensure that non-autonomous ODEs are solved correctly;

(2) when calculating the appropriate value of for the gradient-modifier $m_2(0.5(\vec{f}^n + \vec{f}^{(1)}) \prod_{k\in K^n} \frac{\vec{c}_k^n}{\vec{c}_k^{(1)}}, \vec{c}^n)$, the algorithm must recognise that the ultimate size of the external time-step for the projection is no longer Δt_{is} (the provisional external time-step), but rather (approximately, see below) $m_1(\vec{f}^n, \vec{c}^n)\Delta t_{\text{is}}$—where $\vec{f}^n$ denotes the vector of gradients evaluated at time-level t^n using state-variables $\vec{c}^n$, and $\vec{f}^{(1)}$ denotes gradients evaluated at time $t = t^n + m_1\Delta t_{\text{is}}$ using state-variable values $\vec{c}^{(1)}$;

(3) ultimately, the realised time-step for one cycle of the integration algorithm will be of size

$$\Delta t_{\text{ds}} = m_2\left(0.5\big(\vec{f}^n + \vec{f}^{(1)}\big) \sqrt[k]{\prod_{k\in K^n} \frac{c_k^n}{c_k^{(1)}}}, \vec{c}^n\right) m_1\big(\vec{f}^n, \vec{c}^n\big)\Delta t_{\text{is}}.$$

Four further points need comment:

(1) since

$$\Delta t_{\text{ds}} = m_2\left(0.5\big(\vec{f}^n + \vec{f}^{(1)}\big) \sqrt[k]{\prod_{k\in K^n} \frac{c_k^n}{c_k^{(1)}}}, \vec{c}^n\right) m_1\big(\vec{f}^n, \vec{c}^n\big)\Delta t_{\text{is}} \leqslant m_1\big(\vec{f}^n\big)\Delta t_{\text{is}},$$

m_2 has been calculated on the basis of what will prove to be an over-estimate of the eventual, realised time-step.

(2) More importantly, the second set of gradient estimates $f^{(1)}$ has been made at a time which will prove to be beyond the eventual end of the time-step. In order to ensure that the 'corrector'-stage of the integration is made using time-centred temporal gradients, the relative weights assigned to the first and second estimates of the temporal gradients (i.e. f^n and $f^{(1)}$, must be recalculated during every cycle of the integration algorithm). Despite the fact that $m_2(0.5(\vec{f}^n + \vec{f}^{(1)}) \sqrt[k]{\prod_{k\in K^n} \frac{c_k^n}{c_k^{(1)}}}, \vec{c}^n)$ was calculated assuming an even weighting between the two gradient estimates, the subsequent change to the weighting (giving $\vec{f}^n$ greater weight) can be made without endangering the guaranteed positivity of the ultimate projection—because the ultimate, realized size of the time-step is lower than that which was envisaged when m_2 was calculated. Adopting notation analogous to that used in Press et al. [7], let us define b_1 to be the weight given to $\vec{f}^n$, and b_2 be the weight given to $\vec{f}^{(1)}$ when making the final 'corrector' projection. To maintain time-centring:

$$b_1 = \left\{1 - \frac{m_2}{2}\right\}; \qquad b_2 = (1 - b_1) = \frac{m_2}{2},$$

and

$$\vec{c}^{n+1} = \vec{c}^n + m_1 m_2 \Delta t_{\text{is}}\big(b_1\vec{f}^n + b_2\vec{f}^{(1)}\big),$$
$$t^{n+1} = t^n + m_1 m_2 \Delta t_{\text{is}}.$$

(3) When other components (e.g. advection and dispersion) of the system are to be solved with different numerical methods, one of two things must happen. Either the other numerical methods must perceive Δt_{ds} rather than Δt_{is}(requiring that local-scale reaction dynamics be solved first), or the amBBKS scheme must be cycled until Δt_{is} has been traversed. We note that $m(\vec{f}, \vec{c}) < 1$. Thus, as with the samBBKS scheme, one must either rely upon floating-point-underflow to stop the iteration of the scheme, or introduce a minimum size for the provisional size of an internal time-step, and accept the implication that the nominated external time-step will eventually be slightly more than traversed.

(4) In cases where the amBBKS scheme is cycled only once (implying that the realized size of the external time-step will be less than that initially nominated), it might be tempting to use the realized step size as the provisional size of the next step. This is not possible because sequential provisional external step-sizes would become ever-smaller. If one insists upon choosing a provisional external step size on the basis of the previous realized step-size, one must choose $\Delta t_{\text{is}}^{n+1} = (1 + \mu)\Delta t_{\text{ds}}^n$ with $\mu > 1$. The accuracy of the scheme will tend to fall as μ is increased.

References

[1] N. Broekhuizen, J. Zeldis, J. Oldman, Sub-grid-scale differences between individuals influence simulated phytoplankton production and biomass in a shelf-sea system, Marine Ecology Progress Series 252 (2003) 61–76.
[2] J. Bruggeman, H. Burchard, B. Kooi, B. Sommeijer, A second-order, unconditionally positive, mass-conserving integration scheme for biochemical systems, Applied Numerical Mathematics 57 (1) (2007) 36–58.
[3] H. Buchard, E. Deleersnijder, A. Meister, A high-order conservative Patankar-type discretisation for stiff systems of production–destruction equations, Applied Numerical Mathematics 47 (1) (2003) 1–30.
[4] S.C. Doney, H.W. Ducklow, A decade of synthesis and modeling in the US joint global ocean flux study, Deep Sea Research II 53 (2006) 451–458.
[5] C. Lancelot, E. Hannon, S. Becquevort, C. Veth, H.J.W. De Baar, Modeling phytoplankton blooms and carbon export production in the Southern Ocean: dominant controls by light and iron in the Atlantic sector in Austral spring 1992, Deep Sea Research 47 (9) (2000) 1621–1662.
[6] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, Series in Computation Methods in Mechanics and Thermal Sciences, McGraw-Hill, New York, 1980.
[7] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, Cambridge University Press, Cambridge, 1992.
[8] A. Sandu, Positive numerical integration methods for chemical kinetic systems, Journal of Computation Physics 170 (2) (2001) 589–602.
[9] L.F. Shampine, Conservation laws and the numerical solution of ODEs, Computers and Mathematics with Applications B 12 (5–6) (1986) 1287–1296.