Modelling techniques for biological reaction systems 3. Modelling of the dynamic case

A E Billing and P L Dold*

Department of Chemical Engineering, University of Cape Town. Private Bag, Rondebosch 7700, South Africa.

Abstract

This paper is the last in a series of three which deals with numeric techniques for biological reaction systems. The dynamic problem involves solving a set of coupled ordinary differential equations. A multirate technique based on Gear's approach is developed. The method incorporates a variable steplength facility.

Introduction

In practice the inputs to a biological system are unlikely to remain constant. Because the influent to the system varies with time, the mass balance equations describing the response of the system will take the form of a set of differential equations incorporating time-dependent terms [see Eqs. (14) to (21) of Part 1 in the series, for example.] This set of equations will define how the values of the concentrations of each compound in each reactor (the state variables) vary with time.

Solving the set of simultaneous differential equations is an initial value problem. The magnitudes of the concentrations of each compound in each reactor are specified as the initial condition, and thereafter the equations are solved by integrating forward in time. In this way, the changes in concentration in each reactor can be tracked, subject to the variations in the influent flow rate and concentrations. In certain circumstances, such as an activated sludge system, the influent pattern of flow rate and concentration is repeated closely from day to day i.e. a daily cyclic basis. A useful facility, therefore, is to predict the steady state cyclic response when it is assumed that the influent pattern is repeated identically from day to day. Because the initial values are only approximations, finding this solution will require integrating forward through perhaps many cycles until convergence to the solution is attained. Convergence in this case requires that the cyclic concentration response of each compound in each reactor is identical from cycle to cycle, and the values at the start and end of each cycle are the same.

The set of differential equations describing the response of a biological system under dynamic conditions will contain non-linear terms, as did the mass balance for the steady state case. The task of finding the solution to such a set of non-linear ordinary differential equations is certainly not unique to biological systems. Many systems of interest to engineers and scientists are described by non-linear differential equations. A multitude of numerical integration techniques exists for the solution of these sets of equations. Consequently, when faced with such a set of equations, the problem in finding a numerical method is the selection of an appropriate one from the many diverse methods available.

This paper outlines the selection of an integration scheme appropriate for the dynamics of biological reaction systems. In the selection, the approach taken was to first establish a rudimentary integration module which was then refined and improved. In the process of refining the module, a greater understanding of the actual dynamics of the system was generated. Thus, through an interactive process, the integration routine was gradually tailored to better meet the demands of the biological system under consideration. The chronological development of the integration module is presented here. Information concerning general aspects of integration and the earlier versions of the module are included as the detail facilitates developing a broader understanding of the dynamic problem.

* To whom all correspondence should be addressed.

Received 27 October 1987

General comments on using numerical integration techniques

Because the exact solution to the set of differential equations is not, in general, known and cannot be calculated analytically, a numerical integration technique will be required to provide an approximation to the solution. A common approach, which will be the focus of this presentation, is to use a time-stepping or difference method which approximates the solution by its value at a sequence of discrete points called the mesh points. Given a differential equation $y'(x) = 0$, a difference method provides some rule for approximating y at a point x_n ($y(x_n)$) in terms of the value of y at x_{n-1} and possibly at preceding points. Ideally, the solution should be represented by its actual value at each mesh point so that it can be approximated to high accuracy by interpolation between the mesh points. However, the exact solution to the differential equation is not known, so it is always an approximation that is sought. Many techniques assume that the mesh points are equally spaced. However, since the stepsize seems to have an effect on the error introduced, it is usually possible to vary the mesh spacing to account for this. For the moment, it will be assumed that the mesh spacing remains constant during the stepping procedure.

The simple Euler method

The simplest stepping technique available is Euler's rule. The value of the dependent variable at one point is calculated by straight line extrapolation from the previous point. Consider the function y with

$$y'(x) = \frac{dy}{dx} = f(x,y) \tag{1}$$

The value of y at $x_{n+1} = (x_n + h)$ may be approximated by a Taylor's expansion. Truncating after the first two terms in the series yields:

$$y(x_n + h) \approx y(x_n) + h \cdot f(x_n, y(x_n)) \tag{2}$$

where h = steplength

The error in this approximation is described by the remaining terms in the Taylor's expansion:

$$\frac{h^2}{2!} \cdot y''(x_n) + \frac{h^3}{3!} \cdot y'''(x_n) + \ldots \qquad (3)$$

and is called the **local truncation error.** A more detailed discussion about the errors introduced by a stepping method, and the resultant implications will be covered later.

Euler's rule is usually formulated as:

$$y_{n+1} = y_n + h \cdot f_n \qquad (4)$$

and in this form can be described as an explicit linear one-step method of first order.

Dahlquist and Bjorck (1974) provide an example to illustrate the use of Euler's formula for a single differential equation:

$$\frac{dy}{dx} = y \text{ with } y(0) = 1$$

Euler's rule gives the following:

$$y_{n+1} = y_n + h \cdot y_n \quad y_o = 1 \qquad (5)$$

Table 1 presents the results obtained by first computing the solution with h = 0,2 and then with h = 0,1, and compares these with the exact solution. An examination of the Table reveals that the error is approximately proportional to the stepsize. In other words, if the error in the integration is to be halved, then the stepsize will also need to be halved. This implies that, to attain reasonable accuracy with Euler's method, the stepsize chosen needs to be small. This is an inherent weakness in using a first order method.

Multistep methods and predictor-corrector pairs

Multistep methods present a distinct advantage over one-step methods such as the first order Euler's rule. These methods exhibit improved accuracy and convergence characteristics, although at the expense of requiring additional computation. Recall that Euler's method only required the value at one mesh point to compute the value at the next. Multistep methods use more than one value of the dependent variable to calculate the equivalent information at the next time interval. Recall also that Euler's method was referred to as explicit; that is, y_{n+1} occurs only on the left hand side of the equation and can be calculated directly from the right-hand side values. Linear multistep methods are generally implicit; that is, the unknown value occurs on both sides of the equation and cannot be calculated directly. These implicit methods in general entail a substantially greater computational effort than do explicit methods. On the other hand, implicit methods can be made more accurate than explicit methods and enjoy more favourable stability properties (Lambert, 1974). In fact, these considerations so favour implicit methods that explicit linear multistep methods are seldom used on their own.

The following formula, the second order trapezoidal method,

$$y_{n+1} - y_n = \frac{h}{2} \cdot [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \qquad (6)$$

is an example of an implicit method, since y_{n+1}, which is to be computed, appears implicitly on the right-hand side. If f is a non-linear function, a non-linear system will need to be solved at each step. This must be done by some iterative method, for example, by the procedure:

$$y_{n+1} = \frac{h}{2} \cdot [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] + y_n \qquad (7)$$

To solve Eq. (7), a reasonable initial estimate of y_{n+1} can be obtained using past values of y with, for example, Euler's explicit formula. In this context, the explicit formula is usually referred to as the predictor, whilst the implicit formula of Eq. (6) is referred to as the corrector. Used in combination, these two equations make up a procedure called a predictor-corrector method. Lapidus and Seinfeld (1971) refer to a predictor-corrector method that is used in this way as a PECE method, indicating that a **predicted** value of y_{n+1} is followed by a derivative **evaluation,** y'_{n+1}, and then y_{n+1} is **corrected** and y'_{n+1} **evaluated.**

Termination of the integration step may be controlled in one of two ways. The first consists of continuing the iterative scheme suggested by Eq. (7) until the iterates have converged. In practice, this would usually involve comparing the difference between two successive estimates of the solution to some preset tolerance. Since each iteration corresponds to one application of the corrector, this mode of operation of the predictor-corrector method is referred to as **correcting to convergence** (Lambert, 1974). In this mode, there is no way of telling in advance how many iterations will be necessary and consequently how many function evaluations will be required at each step.

TABLE 1
EULER'S METHOD EXECUTED WITH THE TWO DIFFERENT STEPSIZES (DAHLQUIST AND BJORCK, 1974).

Exact solution		STEPSIZE					
		h = 0,2			h = 0,1		
x_n	$y(x_n)$	y_n	$h \cdot f_n$	error	y_n	$h \cdot f_n$	error
0,0	1,000	1,000	0,200	0,000	1,000	0,100	0,000
0,1	1,105				1,100	1,110	−0,005
0,2	1,221	1,200	0,240	−0,021	1,210	0,121	−0,011
0,3	1,350				1,331	0,133	−0,019
0,4	1,492	1,440	0,288	−0,052	1,464	0,146	−0,028
0,5	1,649				1,610	0,161	−0,039
0,6	1,822	1,728		−0,094	1,771		−0,051

The second approach for terminating the integration step is motivated by the desire to restrict the number of function evaluations per step. The number of times, M, that the corrector is applied at each step is stipulated in advance. This approach is more common than the method of correcting to convergence. This mode of operation can be described as a PE(CE)^M method, where the predicted y_{n+1} and evaluated f_{n+1} is followed by M corrections and derivative evaluations.

Of the two approaches, Lapidus and Seinfeld (1971) recommended the PE(CE)^M method with M = 1 as being one of the most successful means to apply these predictor-corrector formulae. Using only a single application of the corrector formula saves on the number of function evaluations required — only two function evaluations are required per iteration.

Error control

Once a time-stepping method has been selected to carry out the numerical integration procedure, the next stage is to evaluate the accuracy of the solutions that it generates. For each step of the difference procedure, some form of approximation is used to obtain the next estimate of the solution. Thus, each step taken will generate an associated error term. This is a natural consequence of any approximation technique. Given that this error can never be entirely eliminated, the best approach is to ensure that it is continuously evaluated and maintained at acceptable levels. In addition, the stepping procedure should be able to incorporate adjustments to the relevant parameters as soon as the error begins to accumulate. It is the nature of this cumulative error that will be decisive in the eventual success or failure of each step of the integration method.

Sources of error

In using a stepping or difference method to find the solution to a differential equation, the solution that is eventually found will never be exact. The difference between this solution and the exact solution is the local error. Sources of error will include, amongst others. (Dahlquist and Bjorck, 1974):

- the round-off error introduced by using finite precision numbers; and
- the truncation error associated with the linear multistep method used. This is the error occurring when a limiting process is truncated or broken off before the limiting value has been reached.

Once the primary sources of error in a stepping method have been identified, it is necessary to be able to use this information in such a way as to ensure that all errors are minimised as far as possible. This will involve an assessment of how each of the error terms affects the reliability of the method, which sources of error dominate and how a knowledge of the error can be used in maintaining the accuracy of the stepping algorithm.

Estimating the local error

Dahlquist and Bjorck (1974) demonstrate that, for an integration method of order p, the local error is approximately bounded by:

$$l_n \approx \left| c_n \cdot \left[\frac{h}{2} \right]^{(p+1)} \right| \tag{8}$$

where l_n = local error
p = order of the integration method
h = stepsize
c_n = a constant specific to the integration method

Dahlquist and Bjorck (1974) provide an alternative formulation for a predictor-corrector method. They propose that the error of the predicted value can be expressed by a difference function using a constant, c', which is specific to the order of the predictor. The difference between the predicted and the corrected values, multiplied by c/(c' - c) is then an estimate of the local error of the corrected value:

$$l_n = \frac{c}{(c' - c)} \cdot (y^P - y^C) \tag{9}$$

where c = a constant specific to the order of the predictor
c' = a constant specific to the order of the corrector
y^P = predicted value of y
y^C = corrected value of y

Percentage accuracy

An error tolerance must be selected to satisfy the dual requirements of reliability and efficiency. If a very strict tolerance is chosen, unnecessary computational effort will be expended in order to meet its requirements. If the tolerance chosen is not sufficiently stringent, it is possible that the effect of a cumulative error will eventually lead to instability of the method and jeopardise its chances of successful convergence. In practise, it has been found convenient to express this tolerance in terms of a "percentage accuracy" where this percentage accuracy is defined in the same way as a relative error measurement. That is:

$$\%\ \text{acc} = \frac{y_{(p-1)} - y_{(p)}}{y_{(p-1)}} \cdot 100 \tag{10}$$

where $y_{(p)}$ = current estimate of the value of the variable
$y_{(p-1)}$ = previous estimate of the value of the variable
% acc = percentage accuracy

Defining the accuracy requirements in this way enables calculation of error tolerances that are independent of the absolute magnitude of the variables involved. Consequently, the accuracy specifications can be transformed into numerical language that is equally significant for variables with very small or very large magnitudes. This is an important consideration, especially for bad scaling problems, such as those encountered in biological systems. Practically, this is achieved with the use of an error tolerance, $\in$, which is defined in terms of the percentage accuracy required and is the limiting value that the local error may reach without jeopardising the success of the step. Once a percentage accuracy is specified, this must be transformed into an $\in$ value which applies to each variable. This can be achieved by firstly reformulating Eq. (10) to give:

$$\%\ \text{acc} = \frac{y^P - y^C}{y^P} \cdot 100 \tag{11}$$

Given that, in the limiting situation,

$$\in = l_n \tag{12}$$

a limiting value for $\in$ for each variable in a set of simultaneous diffential equations may be derived by combining Eqs. (9), (11) and (12) as follows:

$$\in = \frac{c}{(c' - c)} \cdot (y^P - y^C)$$

$$= \frac{c}{(c' - c)} \cdot \frac{\% \text{ acc} \cdot y^P}{100} \qquad (13)$$

Eq. (13) now provides a means of selecting an error tolerance for each variable which is based on a percentage accuracy requirement but which also incorporates a measure of the scale of the variable involved, y^P. Eq. (13) is useful because it allows the user to specify a completely general percentage accuracy requirement for the integration module. This specification is then used to calculate a separate $\in$ value for each component in the system. This error tolerance can now be used as the basis for estimating the next stepsize for each component.

Stepsize selection

For any efficient difference method for integration, an objective is to use integration steps that are as large as possible, whilst preserving the required accuracy. Once an estimate has been made of the magnitude of the error generated at an integration step, this estimate can be used to decide whether or not the most recently computed value of the variable is acceptable. If the error is found to be larger than a predetermined tolerance, the value will be rejected and then recomputed using a smaller stepsize. If the error is within the bounds prescribed, then the value will be accepted and a larger step can be taken, in order to generate a new estimate.

Ideally, the size of each new step should be selected so that it reflects the magnitude of the error in the previous calculation. In other words, if the value falls well within the prescribed error bounds, a large increase in the steplength should be permitted. If the error in the value is close to the tolerance, then the subsequent steplength should be allowed to increase, but not so dramatically. It is suggested by Dahlquist and Bjorck (1974) that, in order to maintain the local error below a given tolerance, the new stepsize should satisfy the following condition:

$$c_n \cdot \left[\frac{h'}{2}\right]^{(p+1)} \leqq \Theta \cdot \in \qquad (14)$$

where h' = new stepsize
Θ = a preset safety factor to account for the fact that the error estimates are approximate and based on experience from the preceding interval ($\Theta \leqq 1$)

From Eq. (9):

$$l_n \approx c_n \cdot \left[\frac{h}{2}\right]^{(p+1)} \qquad (15)$$

Therefore, eliminating c_n in Eq. (14) yields:

$$h' = \cdot \left[\frac{\Theta \cdot \in}{l_n}\right]^{1/(p+1)} \qquad (16)$$

The usefulness of this formulation to determine the subsequent stepsize rests on the fact that it incorporates the absolute magnitude of the error generated by the previous step. This means that the calculation of the next stepsize is based on a quantitative assessment of exactly how successful the previous step was.

Dynamic behaviour of biological systems

The preceding sections have dealt with the selection of integration technique and steplength adjustment in general terms. This information is now implemented for the simulation of biological system behaviour under dynamic conditions. The discussion is best introduced by considering a numerical example.

Consider the behaviour observed in an aerated batch reactor into which heterotrophic organisms (X_B) and a readily biodegradable soluble substrate (S_S) are introduced at time t = 0. Assume that the initial concentrations are X_{BO} = 1 000 g.m^{-3} and S_{SO} = 100 g.m^{-3} respectively. Assume also that the behaviour in the batch reactor is governed by the model introduced earlier (Part 1 in the series) and that the kinetic and stoichiometric constants are those used in the case studies (Part 2, Table 1). At the start of the batch test, the changes in concentration of X_B and S_S will be dominated by the growth process. Organism decay will exert only a minor influence on X_B. The rates of change of concentration at t = 0 will be:

For X_B:

$$\left.\frac{dX_B}{dt}\right|_O = \hat{\mu} \cdot \frac{S_{SO}}{(K_S + S_{SO})} X_{BO} - b \cdot X_{BO}$$

$$= 4 \cdot \frac{100}{(5 + 100)} \cdot 1\,000 - 0{,}62 \cdot 1\,000$$

$$= 3\,189{,}5 \text{ g}\cdot\text{m}^{-3} \cdot \text{d}^{-1}$$

For S_S:

$$\left.\frac{dS_S}{dt}\right|_O = -\frac{\hat{\mu}}{Y} \cdot \frac{S_{SO}}{(K_S + S_{SO})} X_{BO}$$

$$= \frac{-4}{0{,}666} \cdot \frac{100}{(5 + 100)} \cdot 1\,000$$

$$= -5\,720{,}0 \text{ g}\cdot\text{m}^{-3} \cdot \text{d}^{-1}$$

If these rates were to persist unchanged, then the S_S concentration would be reduced by 100 per cent to zero after a period of 25 min; at this time the concentration of X_B would be 1 055,7 g·m^{-3} i.e. only 5,5 per cent greater than the initial value. In practice, this would not occur, as the growth rate decreases with decreasing S_S concentration, particularly once the S_S concentration falls below 10 g·m^{-3}. The actual progression of the batch test over the initial period would be as shown in Fig. 1.

Let us now consider simulation of the batch test behaviour. If, for example, the Euler rule were employed and a steplength of

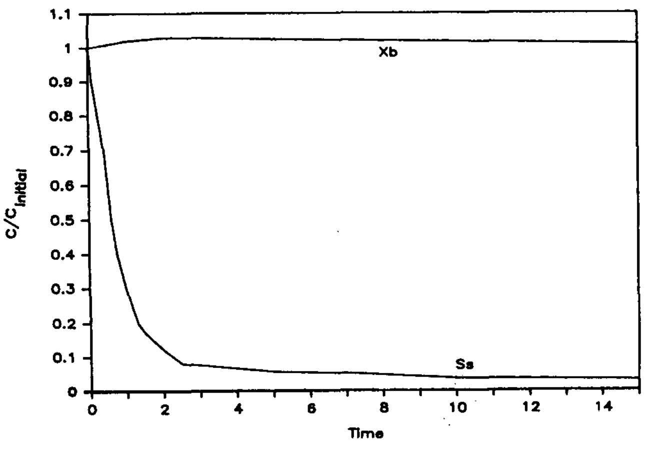

Figure 1
The progression of a batch test showing the response of S_S and X_B.

30 min (0,5 h) were used, then the predicted concentration for S_S at t = 0,5 h would be:

$$\begin{aligned} S_S &= S_{SO} + \left.\frac{dS_S}{dt}\right|_0 \Delta t \\ &= 100 - 5\,720{,}0 \cdot \frac{0{,}5}{24} \\ &= -19{,}2\ \text{g} \cdot \text{m}^{-3} \end{aligned}$$

Clearly, this result is meaningless and much shorter steplengths would be required, perhaps of the order of 1 min. In this case, after the first minute, the predicted concentrations of S_S and X_B would be:

$$\begin{aligned} S_S &= 100 + (-5\,720{,}0) \cdot \frac{1/60}{24} \\ &= 96{,}0\ \text{g} \cdot \text{m}^{-3} \text{ (i.e. a change of 4 per cent)} \\ X_B &= 1\,000 + 3\,189{,}5 \cdot \frac{1/60}{24} \\ &= 1\,002{,}2 \text{ (i.e. a change of 0,22 per cent)} \end{aligned}$$

Given that the percentage change of X_B is only one twentieth that of S_S over the interval, it appears that the steplength of 1 min is unnecessarily short to track the changes in X_B with acceptable accuracy. However, because the variables X_B and S_S are coupled, the equations should strictly be integrated simultaneously. This implies that the steplength used for the integration procedure will be limited by the maximum allowable size for the rapidly changing S_S, and X_B will be tracked with "unnecessary" accuracy.

Gear (1984) noted that behaviour similar to the response in the batch test is encountered in many engineering systems. Although strictly these variables are coupled, he suggested that the degree of coupling between the variables might not be strong. Gear proposed that, if this is so, then the differential equations for each of the variables may be integrated separately. In the batch reactor example, Gear's approach would mean that longer steplengths could be used for integrating X_B than those required for S_S. This offers the advantage of increased computational efficiency without compromising on accuracy requirements.

Implementing Gear's multirate approach involves partitioning a system of equations into different groups, each of which is governed by different dynamics. A group governed by "fast" dynamics would require short integration steps, whereas a group exhibiting "slow" dynamics could be integrated using longer integration steps. Gear proposed a number of schemes to account for the coupling between components with "fast" and "slow" dynamics.

The use of a multirate technique

Methods for handling the coupled equations

Consider a two-component system (one fast, one slow) which is described by two coupled ordinary differential equations. Assume that this system is integrated from t_0 to $(t_0 + \Delta t)$ using a stepsize h for the "fast" component and H for the "slow" component, where $H > h$ and $H = rh$. This division is shown in Fig. 2.

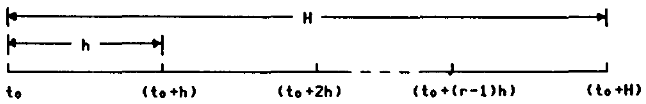

Figure 2
Schematic representation of small and large timesteps for a multirate integration technique.

At any point in the integration process, values of both the "slow" and the "fast" variables will be required in order to complete the next integration step. At $(t_0 + h)$, the next integration step for the "fast" component to $(t_0 + 2h)$ will require a knowledge of the value of the "slow" compound at least at $(t_0 + h)$. If the "slow" compound is integrated first, then its most recently computed value will be that at the end of the long time interval, $t_0 + H$. If the "slow" compound has not yet been integrated, then the latest available value will be that at t_0, the beginning of the large time interval. In either case, an explicit estimate of the value of the "slow" compound at $(t_0 + h)$ is not available. The same problem will be encountered at $(t_0 + 2h)$, $(t_0 + 3h)$, . . . etc. Estimation of the value of the "slow" compound at these intermediate points $(t_0 + h,\ t_0 + 2h,\ \ldots,\ t_0 + (r-1)h)$ is the crux of the problem of simultaneous multirate integration.

Two approaches to solving the problem have been suggested. The first approach ("fastest first") integrates the "fast" compounds over $(r-1)$ steps of size h and then simultaneously integrates the "fast" and "slow" components to advance them to the end of the interval using stepsizes h and H respectively. For the small integration steps, values of the "slow" component are obtained by extrapolation from previous values (i.e. at t_0 and before) with a predictor-like formula. Gear (1984) notes that the error incorporated into the method through this extrapolation should be "of tolerable size". The disadvantage of this "fastest first" approach, however, is the amount of storage space consumed by the necessity to back up the "slow" variables in case of an integration step failure. If a step fails, and the "slow" variables have not been stored, then they will need to be recalculated. The effort involved in accomplishing this task is an additional drawback to the "fastest first" approach.

The second approach ("slowest first"), which has been recommended by Gear (1984) and is adopted in this study, involves integrating the "slow" compounds to the end of the largest timestep first. Interpolation techniques are then used to obtain intermediate values for the "slow" compounds at the points $(t_0 + h)$, $(t_0 + 2h)$, . . ., $(t_0 + (r - 1)h)$. This then allows integration of the "fast" compounds in short steps from t_0 to $(t_0 + H)$. With this approach, the extrapolation of the "fast" variables to integrate the "slow" variables first will lead to large errors in the extrapolated values because the extrapolation is over many timesteps in the "fast" variables. Gear maintains that this is mitigated by the fact that coupling from the "fast" values to the "slow" values is generally small. This, in fact, is the basis for the method. An important advantage of the "slowest first" method is that, if a variable has to be backed up because of an integration failure in another variable, the backup is simply a reduction of the size of the last step taken, and can be done provided that only one additional value is kept for all variables.

Partitioning of a system

Partitioning a system into categories having different dynamics is an important consideration in the use of a multirate technique. Various methods for automatic partitioning of the system have been investigated (Gear, 1984; Orailoglu, 1983). These methods have been found difficult to implement and expensive in terms of computational time. Therefore, static partitioning, where the division into categories is specified by the user prior to the integration, is generally applied.

A problem that arises in partitioning is what exactly defines dynamics as "fast" or "slow"? One possible answer is that "fast" components exhibit a large differential term i.e. (dC/dt) is large. Another possible solution would be that it is a rapidly changing (dC/dt) term that indicates "fast" dynamics. Alternatively, perhaps the ratio $(dC/dt) / C$ is an appropriate measure for reaction systems where the concentration C cannot decrease to less than zero. The question does not appear to have been resolved in the literature and decisions as to the classification of compounds are usually based on practical experience and knowledge of the physical system.

Integration errors with a multirate technique

In choosing a time-stepping algorithm for the solution of the dynamic problem, the question of error estimation and control is a central one. Decisions as to whether or not concentration variables are acceptable have to be based on some estimation of how close these are to the actual solution. In using a multirate technique to carry out the integration procedure, the contribution to the global error in the method may stem from a number sources. For the "slowest first" technique, the sources of error include the round-off error and the local truncation error described earlier, as well as:

- The error associated with the extrapolation of the "fast" components to allow integration of the "slow" components even though the fast components have yet to be integrated.
- The error associated with the interpolation of the slower components to allow integration of the "fast" components. The interpolation error depends on the method used, and is due to two effects: firstly, the errors in the interpolation formula itself and secondly, errors due to errors in the mesh values. Gear (1984) notes that the errors in the interpolation formula will be significantly less than those in an extrapolation formula over the same interval.

In controlling the error, Gear (1984) suggests ensuring that the contributions from the interpolation and extrapolation are small in comparison to the local truncation error term. The local error term can then be used to select the next stepsize on the basis of a given error tolerance. An error tolerance term, $\in$, can be calculated for each component in each of the "fast" and "slow" groups. Once all the $\in$ values for all the components have been computed, the largest $\in$ value for each group is chosen as the limiting value. It is this limiting $\in$ that will determine the size of the subsequent integration step for all the components in that group.

Stepsize selection with a multirate technique

The concept of using a multirate method is to reduce computation in an integration problem by using different stepsizes for groups of components with differing dynamics. Efficiency can be increased further by using the longest possible stepsize within each group.

Gear (1984) recommends an incremental approach for integration steplength adjustment that was used in the early stages of the development of the integration module here. Very simply, if the error in any component in either the "fast" or "slow" group is greater than the prescribed tolerance, then the value is rejected and the next steplength for that group will be half the size of the previous one. If the error is less than the prescribed tolerance, then doubling of the next steplength is permitted. These stepsize changes are subject to certain constraints and may only take place at particular points in the integration scheme. This is possibly the most simple approach that can be taken in formulating some kind of dynamic relationship between error magnitude and stepsize control. One of its major limitations is that it makes no distinction between solution estimates that fall well within the prescribed error bounds and those that only just satisfy the error criterion. This is a significant limitation because the stability behaviour of the system is detrimentally affected by the accumulation of errors in the calculated variables.

A possible approach to overcome the limitations of Gear's method for steplength adjustment, would be to use the variable steplength adjustment method of Dahlquist and Bjorck (1974) discussed earlier. This is justified because the local error term is dominant.

Implementation of Gear's multirate technique

The basis for developing and evaluating the multirate integration technique was a continuation of the case study introduced in Paper 1; that is, the single reactor plus settling tank problem based on the limited IAWPRC model. Eqs. (14) to (21) of Part 1 are the set of differential equations and algebraic equations describing the system. Numerical values for the problem (reactor volumes, kinetic constants, etc.) were the same as those used for Case Study 1 in Part 2 (Tables 1 and 2).

To introduce the dynamic component, a square wave cyclic input pattern was imposed on the system. In this scheme, the full volume of feed for Case Study 1 in Part 2 was introduced into the reactor at a constant rate but over a twelve-hour period in a twenty-four-hour cycle. For the remaining twelve hours of the cycle, there was no feed. That is, the system was subjected to a step increase in flow rate and twelve hours later to a step decrease to zero flow. This input pattern was selected because, in the region of the step changes, it would provide a rigorous test of the integration method.

There are two requirements before a multirate integration technique can be initiated:

- For an initial value problem such as this, a set of values for each of the state variables at time $t = 0$ is required to initiate the integration. In this case, the simulation program uses the set of state variables which constitute the solution to the steady state problem as initial values for the dynamic case.
- Implementing a multirate technique involves partitioning the compounds into categories with either "fast" or "slow" dynamics. The model incorporates four compounds: three particulate compounds (X_B, X_E and X_S) and one soluble compound (S_S). After investigating the nature of the dynamics of each of these compounds, it was decided to classify X_B and X_E as "slow" and S_S as "fast".
  In the case of X_S, it was found by trial that the dynamics are neither as "fast" as those of the soluble compound nor as "slow" as those of the particulate compounds. In fact, it appears that, in certain circumstances, the behaviour of X_S changes from "fast" to "slow". Classifying the dynamics of X_S as "slow" and using long timesteps for its integration may result in inaccuracies in the solution. On the other hand, categorising it with the "fast" compounds in the system could result in needless extra computational effort as a result of the unnecessarily small timesteps being used at times when X_S exhibits "slow" dynamics. As a result, there seemed to be the potential to incorporate this compound into some kind of intermediate category. Creating an additional category for "intermediate" dynamics would thus have the advantage of enabling the routine to cater specifically for this compound and select an exactly appropriate stepsize for its integration. One drawback of this approach, however, would be the extra programming code required to extend the number of categories from two to three. If the whole purpose of a multirate technique is to improve efficiency, then the added complexity of accounting for the coupling between three groups of variables would perhaps negate this objective at the outset. On the other hand, the ultimate efficiency of the technique depends on the appropriate partitioning of the system. In viewing these alternatives, it was decided to maintain the simpler approach and restrict the number of divisions to two. The effect of partitioning X_S with either the "fast" or the "slow" group is discussed later.

The initial multirate scheme

Gear (1984) has recommended that as many of the parameters as possible in a modern program code should be selected automatically. Achieving this for an integration scheme does present some difficulties particularly when implementing a multirate technique. This is due to the large number of parameter choices involved. As a result, the initial approach to the problem relied on prior specification of a number of the variables in accordance with the suggestions of Gear (1984):

- The initial stepsize for the "slow" components, H, was set at an arbitrary value to initiate the integration.
- The ratio of the number of small steps to the number of large steps was also specified and remained fixed throughout the integration.
- Initially, the steplengths could only be increased or decreased by a factor of two.

The "slowest first" technique recommended by Gear (1984) was followed. A simple Euler formula was used for both the "slow" and the "fast" integrations.

An obvious limitation of this initial approach is the fact that the error is checked only at the end of every large time interval. If the errors in all the compounds are acceptable, only then can the large timestep be doubled. If the error in any one of the compounds is not within the limits prescribed, then the large timestep is halved. Since the ratio of the number of "fast" to the number of "slow" timesteps remains constant, halving the size of the large timesteps also means halving the size of the small timesteps which may not be necessary.

The manner in which steplengths were adjusted is a major inefficiency in the method. In practice, the error in the "fast" compounds was found to be both larger and to accumulate more rapidly than that in the "slow" compounds. Allowing the error in the "fast" compound to accumulate until the end of the large timestep, besides being inefficient, also very often upset the success of the step. In addition, with a fixed ratio of small to large timesteps, the size of the large timestep often had to be unnecessarily reduced in order for the small timestep to be successful.

For this particular system, it was found that if an error in any component was permitted to approach the error bound, it then began to accumulate very rapidly. Eventually this affected the stability of the entire system. Consequently, the integration module needs to be formulated in such a way that errors could be evaluated as soon as any timestep, "fast" or "slow", had been completed. In addition, corrective action should be taken immediately. The cost of such an evaluation process was considered to be well worth it, as it was critical to the stability of the entire system.

An improved version

In an attempt to overcome the limitations outlined above, two refinements were incorporated in the algorithm:

- Allowance was made for the small timestep ("fast" dynamics) to vary independently of the large timestep ("slow" dynamics). This replaced the scheme of having a fixed number of "fast" steps per "slow" step. The error in the "fast" step was now evaluated immediately, and the short steplength doubled or halved as appropriate. With both the "slow" (large) and "fast" (small) timesteps being variable, full advantage is taken of the different dynamics of the system. Appropriate action is taken as soon as the error reaches unacceptable proportions. This improved version implies that the groups of "fast" and "slow" compounds are being integrated independently, which is correct, as their different dynamics suggest that they are only weakly coupled.
  A restriction on the step adjustment procedure was that step doubling could only take place at a synchronisation point in the mesh. This was in accordance with the suggestion of Gear (1984), who motivated that synchronisation of the "fast" and "slow" meshes is desirable to prevent unnecessary interpolations. In this synchronisation scheme, Gear recommends that halving of a short step may take place at any time, but it may only be doubled when $(t - t_0)/h$ is an even number, where h is the current stepsize. If this doubling procedure is followed, then the end of a "fast" integration step will never fall beyond the end of the "slow" step i.e. the steps will be synchronised at the end of the "slow" step. This scheme requires that $(t - t_0)/h$ is an integer.
- The simple Euler rule was replaced by a predictor-corrector pair. The Euler formula was retained as the predictor, and the second order trapezoidal rule was used as the corrector. With this approach, the number of function evaluations would be doubled at each integration step. However, it was hoped that

the more sophisticated integration technique would allow more than a doubling of the steplengths, thus giving an overall increase in efficiency. The single application of the corrector was in line with the recommendation of Lapidus and Seinfeld (1971).

Two problems were apparent with this improved scheme. Firstly, with both timesteps being variable, synchronisation of the meshes is more difficult. Secondly, computational effort can be wasted if the size of the error in the large timestep is not within acceptable limits and the step fails. The error in the "slow" compounds is only checked at the end of every large interval, which means that the already completed computation for the "fast" compounds is wasted if the large timestep is unsuccessful.

To address these deficiencies, two additional modifications were proposed:

- The first improvement in the integration method involved removing the synchronisation constraint. Thus, doubling and halving of "fast" and "slow" steplengths could take place at any point. In the case of the "fast" steps, if by doubling a "fast" step, it was found that the integration would move to beyond the end of the current "slow" step, then a smaller step would be taken to arrive exactly at the end of the "slow" step i.e. truncating to ensure synchronisation. At the start of the next "slow" integration step, the new "fast" steplength would be based on the steplength calculated prior to truncation. This ensured relatively unlimited adjustment of the "fast" steplengths within the "slow" steps. In the case of the "slow" steps, truncation was only required where a "slow" step beyond a data storage point was attempted. In the new scheme, the "slow" step was truncated in a similar manner as for the "fast" step, to end at the data storage point.
- With the improved method, the problem of computational effort "wasted" on the "fast" steps when the "slow" steps failed still existed. In an attempt to overcome the wasted effort, a scheme of multiple corrections was introduced into the integration routine. In this scheme, Euler's formula was used to predict a value for the "slow" compound at the end of the large timestep. The second order trapezoidal rule was implemented as a corrector for the "slow" compounds as before. If the error in the "slow" compounds at the end of the interval was found to be unacceptable, then the corrector was applied again in an attempt to improve the values and reduce the error to within the tolerance. This procedure was motivated by the fact that each correction offers the possibility that the new estimate might be a sufficient improvement to obviate the necessity to halve the steplength and re-perform the calculations. Up to five corrections were applied before the step was abandoned, and the "slow" steplength reduced i.e. from $PE(CE)^1$ to $PE(CE)^5$.

In practice, the multiple correction procedure was not helpful. It was found that the first application of the corrector gave a significant improvement on the value predicted by the Euler rule. However, with repeated applications of the corrector, the improvement was small and the rate of convergence was very slow — no benefit was derived in terms of efficiency.

A modified version of Gear's multirate method

At this stage, a thorough assessment of the integration routine based on Gear's approach illuminated a major deficiency in its operation. This was the fact that it did not account specifically for the range of magnitudes of the error generated at each timestep. The size of each new steplength was only based on whether the value generated at the previous step had satisfied or not satisfied the error criterion. Errors that only just satisfied the error tolerance were accepted and the following steplength was allowed to double, where it would have been more appropriate to increase the steplength by only a small amount. Doubling in this case caused the error to accumulate and a subsequent step would then fail.

The approach suggested by Dahlquist and Bjorck (1974) was used to develop a more sophisticated algorithm which adjusted the steplengths in a manner based on the absolute magnitude of the error generated at the previous step. For the selected predictor-corrector method (Euler/trapezoidal), Eq. (13) and Eq. (16) were used to calculate the size of each new steplength. The relevant constants for the Euler predictor and the trapezoidal corrector are c = 2 and c' = 12, respectively (Lambert, 1974). Thus, Eq. (13) becomes

$$\in = \frac{2}{(12-2)} \cdot \frac{\%\ acc \cdot y^P}{100}$$

$$= \frac{\%\ acc \cdot y^P}{500} \tag{17}$$

Substituting in Eq. (16):

$$h' = h \cdot \left[\frac{\Theta \cdot \%\ acc \cdot y^P}{500 \cdot l_n}\right]^{1/(p+1)} \tag{18}$$

where p = order of the method = 2
l_n = local error (from Eq. (9))

After selecting appropriate values for Θ and % acc, it is now possible to calculate a new stepsize, h', in such a way that it is appropriate to the magnitude of the error generated at the previous stepsize, h.

The algorithm for the final integration module is presented in Appendix A.

Selection of integration parameters

Successful operation of the final multirate method was found to be strongly influenced by the values specified for the parameters of percentage accuracy (% acc) and the safety factor (Θ). The integration routine thus incorporated a facility for these parameters to be selected by the user according to the specific requirements of the system being analysed.

The effect of percentage accuracy

Once the accuracy requirement has been specified, the integration routine uses this value to calculate the limiting value of the local error [$\in$ in Eq. (13)]. Since $\in$ controls the selection of subsequent stepsizes, it exerts a significant effect on the computational effort required to perform the integration.

Fig. 3 shows the effect of different accuracy specifications on the behaviour of the "fast" variable S_S over a typical integration period of an hour when the input to the system is held constant and the integration is initiated at the solution i.e. the values of S_S should remain constant. Three different accuracy requirements were tested: 1,0%, 0,1% and 0,01%. The results are presented in Table 2.

When the percentage accuracy was specified as 1,0% the size of the small timestep was, on average, 12 min long. Seven in-

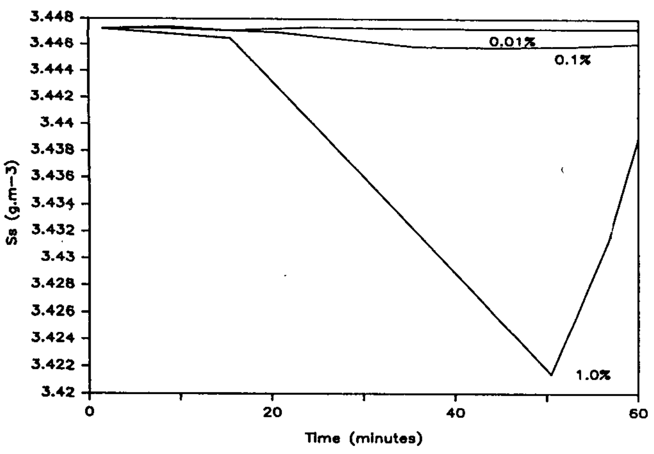

Figure 3
The effect of accuracy specifications on the behaviour of the variable S_S.

TABLE 2
THE EFFECT OF ACCURACY SPECIFICATIONS ON THE BEHAVIOUR OF THE STEPPING ROUTINE.

% accuracy	Number of steps	Number of failures	Average stepsize
0,01%	11	1	8,1 min
0,10 %	10	3	10,6 min
1,00 %	7	2	12,0 min

tegration steps were necessary to reach the end of the interval. However, this includes two steps that were rejected when the accuracy requirements were not met. In addition, the response of S_S was unstable, oscillating more erratically as the timesteps became larger.

When the accuracy requirement was specified as 0,01%, eleven steps were necessary to reach the end of the time interval, the average steplength being 8 min. Only one of the steps failed to satisfy the error tolerance and the response of the variable S_S remained stable at all times. The price paid for the stability of the solution response is the necessity to use small steplengths throughout the integration and thus increase the computational effort expended. Examination of Fig. 3 shows that, for this case, an accuracy requirement of 0,1% appears to meet the demands of stability whilst at the same time not requiring an excessive amount of computational effort, the average steplength being 10,6 min, with a very small oscillation in the response of S_S.

The effect of the safety factor, Θ

Dahlquist and Bjorck (1974) recommend using a safety factor, $\Theta \approx 0,8$ to account for the fact that error estimates are only approximations. To examine how the specification of this factor affected the integration, three different values for Θ were selected and tested in the integration problem outlined above. Fig. 4 shows the effect of the choice of Θ on the stepsizes permitted. When a small magnitude for Θ of 0,5 was specified, steplengths greater than 16 min were never permitted, and were generally much shorter. The average stepsize for the integration interval was 8,3 min and 9 steps were required to reach the end of the interval. Only one of these was unsuccessful (Table 3).

On the other hand, when a large Θ of 0,9 was specified, steplengths were generally longer (average length 10,3 min) with a largest steplength of 19,3 min. However, of the eleven steps required to reach the end of the interval, four were unsuccessful. In the light of this, it would appear that some intermediate value of Θ would offer the most favourable balance between the number of steps required to complete the integration and the possibility of each of these steps being successful. For the purposes of simulation, a Θ value of 0,75 was selected as fulfilling these requirements most appropriately.

TABLE 3
THE EFFECT OF THE SAFETY FACTOR ON THE BEHAVIOUR OF THE STEPPING ROUTINE.

Safety factor (Θ)	Number of steps	Number of failures	Average stepsize
0,50	9	1	8,3 min
0,75	10	3	10,6 min
0,90	11	4	10,3 min

Final comments on partitioning in the multirate method

The effect of X_S as a "fast" or "slow" component

As noted earlier, the dynamics of X_S, the particulate substrate, were difficult to classify as either "fast" or "slow", and there was a general indication that this compound should occupy some "intermediate" category. However, as the creation of an additional class of compounds was not feasible, the effect of placing this compound in either the "fast" or "slow" categories was examined.

When X_S was classified as a "fast" instead of a "slow" component, the sizes of both the "fast" and "slow" steps remained unaffected. In the case of the "fast" steps, this is to be expected, as it is S_S that is the "limiting" compound in the category and which exerts the dominating influence over stepsize selection. Classifying X_S as a "fast" component, however, means that computational requirements are increased for this group, as X_S is now being integrated using many small timesteps. That the size of the "slow" steps did not increase when X_S was removed from the group indicates in fact that X_S is appropriately grouped with the "slow" compounds. This was because, even when X_S was included in the "slow" category, it was observed that the errors in all the "slow" components were consistently small enough to enable the largest possible stepsize to be taken for each integration interval (i.e. H = data storage interval).

From a number of simulations it was found that, even though the dynamics of X_S were "faster" than those of the other compounds in the "slow" category, they were not sufficiently different to cause the error in the integration to increase significantly. As such, when errors at the end of each interval were evaluated, the error in the component X_S was still sufficiently small to allow the largest possible stepsize for subsequent integration steps.

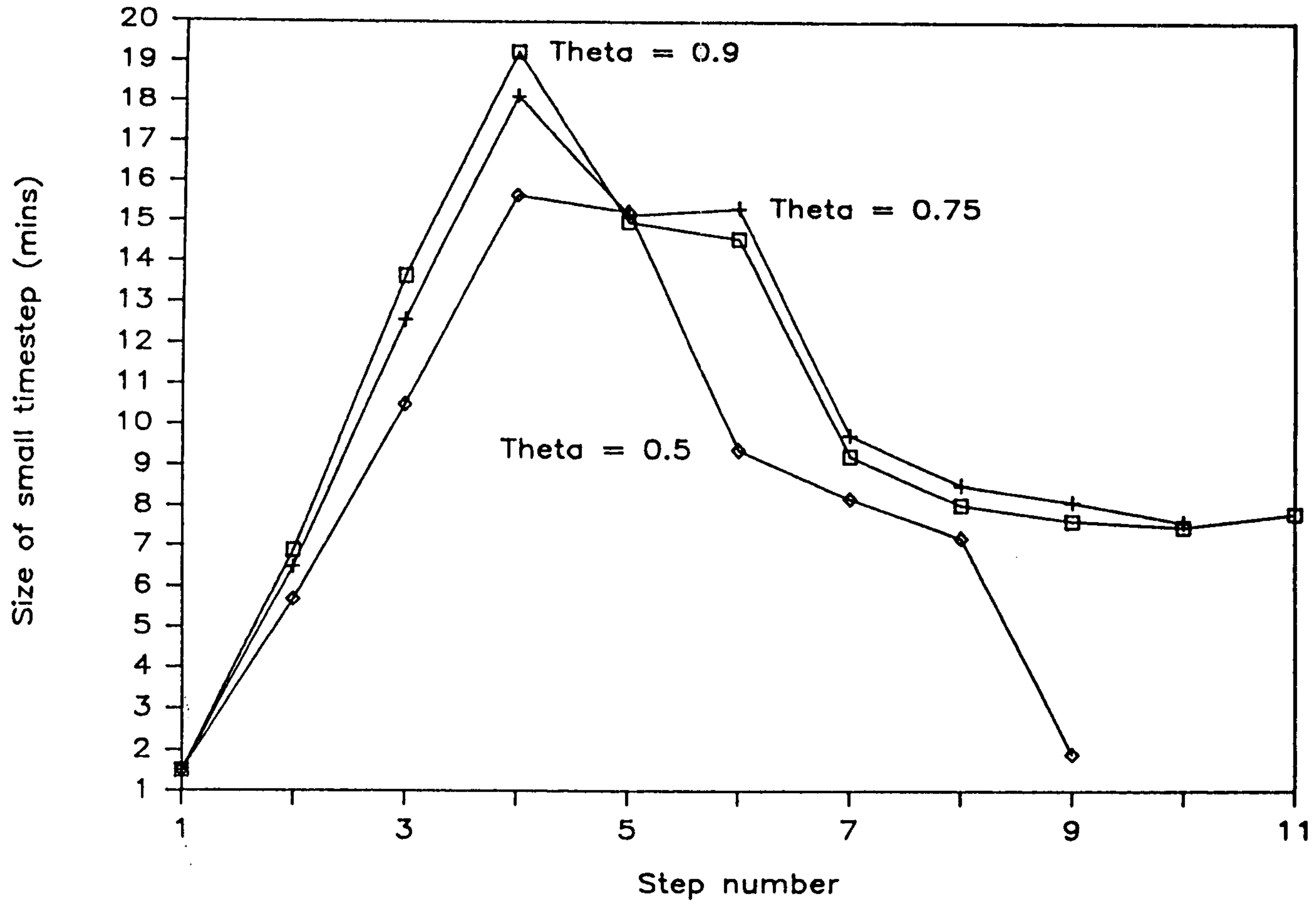

Figure 4
The effect of the safety factor on the size of the small timestep.

A general comment on partitioning

Partitioning of the biological model has been done by comparing the dynamics of each compound to the other compounds in the model. Generally, this led to a division into soluble as "fast" and particulate as "slow" components. A limitation with this approach, which is general to the multirate method, was identified when simulating behaviour in systems with more than one reactor where there were large differences in reactor size. To illustrate the problem, consider the selector reactor configuration of Case Study 2 in Part 2 of this series. In this configuration, the first reactor volume was 1/32 that of the second. Obviously, shorter stepsizes for the "slow" compounds are required in the small reactor with the short retention time than in the larger reactor. Similarly for the "fast" compounds in each reactor. However, because one steplength is chosen for each group, the choice of the stepsize is controlled by the variables in the small reactor. This steplength may be unnecessarily small for the compounds in the larger reactor. In fact, the situation could be encountered where, for optimal multirate efficiency, stepsizes for "fast" compounds in large reactors should be larger than stepsizes for "slow" compounds in small reactors. This problem arises because partitioning is on the basis of the model and not on the basis of individual compounds within the configuration of interest.

The limitation above could be overcome if automatic or dynamic partitioning were implemented. However, this possibility has already been excluded and the limitation had to be accepted. On the other hand, it was felt that generating the division on the basis of the biological model generalised the method and simplified its implementation considerably.

Conclusions

A multirate integration procedure has been found to be appropriate for biological systems. In these systems, the dynamics of the different compounds clearly divide into two groups, a "fast" group requiring short integration steps and a "slow" group for which the steps can be larger. Within each group, the range of dynamics is small compared to the difference between the two groups. A general guideline for partitioning a biological system is that it appears that soluble compounds can be grouped as "fast" and particulate compounds as "slow".

Some specific considerations should be noted as regards implementation of the multirate technique:

- Partitioning of the system on the basis of the model alone can lead to inefficiencies in the implementation of the multirate technique. This is one of the few drawbacks of the method. In practice, partitioning on the basis of practical experience and by trial examination of the system was found to be the most flexible approach.

- A predictor-corrector method with only one application of the corrector as proposed by Lapidus and Seinfeld (1971) was found to be particularly suited to the dynamics of the system.
- The steplength adjustment procedure proposed by Dahlquist and Bjorck (1974) has been found to be appropriate. The success of this procedure rests on the fact that it always uses the largest possible stepsize without allowing the errors to accumulate in the system. The method was found to be superior to Gear's method of steplength halving or doubling.
- Discretion should be exercised in the choice of parameters such as percentage accuracy and the safety factor, Θ.

Appendix A

The final multirate integration algorithm

STAGE 1: Select the following parameters:
(i) initial values for the state variables at $t = 0$
(ii) an initial stepsize, H, for the "slow" components
(iii) an initial stepsize, h, for the "fast" components
(iv) a percentage accuracy requirement, % acc
(v) a value of the safety factor, Θ
(vi) an integration interval for data storage
(vii) a stopping criterion for the 24 h cycle

STAGE 2: For each of the "fast" and "slow" components, calculate $\in$ from Eq. (13), using the initial values of the state variables as the predicted values, y^P:

$$\in = \frac{1}{5} \cdot \frac{\% \text{ acc} \cdot y^P}{100}$$

For the "slow" compounds

STEP 1: Starting at $T = t_0$ and using the Euler formula, compute values for the "slow" compounds at $T = t_0 + H$:
$$y(t_0 + H) = y(t_0) + H \cdot y'(t_0)$$

For the "fast" compounds

Step 1: Starting at $t = t_0$ and using the Euler formula, compute values for the "fast" compounds at $t = t_0 + h$:
$$y(t_0 + h) = y(t_0) + h \cdot y'(t_0)$$

Step 2: Using straight line interpolation, find values for the "slow" compounds at $(t_0 + h)$:
$$y(t_0 + h) = y(t_0) + y'(t_0) \cdot (t - t_0)$$

Step 3: Starting at $t = t_0$ and using the trapezoidal rule, compute corrected values for the "fast" compounds at $t = t_0 + h$:
$$y(t_0 + h) = y(t_0) + \frac{h}{2} \cdot (y'(t_0) + y'(t_0 + h))$$

Step 4: Calculate an error term for each of the "fast" components at $(t_0 + h)$ using Eq. (9):
Set Error $= l_n / \in$

Step 5: Find the largest value of the error term for the "fast" components and use this to calculate the size of the next step using Eq. (16):

$$h' = h \cdot \left[\frac{\Theta \cdot \in}{l_n}\right]^{1/3} = h \cdot \left[\frac{\Theta}{\text{Error}}\right]^{1/3}$$

Step 6: If, by taking this step, the end of the large timestep will not be reached, then
(i) replace h by h'
(ii) replace t by t + h
(iii) return to *Step 1* for the "fast" components.
If, by taking this step, the integration moves to beyond the end of the large timestep, H, then replace h by H − t to arrive exactly at the end of the slow interval, H.
Having integrated to H for the fast compounds, continue to *STEP 2* for the "slow" compounds.

STEP 2: Starting at $T = t_0$ and using the trapezoidal rule, compute corrected values for the "slow" compounds at $T = t_0 + H$:
$$y(t_0 + H) = y(t_0) + \frac{H}{2} \cdot (y'(t_0) + y'(t_0 + H))\text{-}$$

STEP 3: Calculate an error term for each of the "slow" components at $(t_0 + H)$ using Eq. (9):
Set Error $= l_n / \in$

STEP 4: Find the largest value of the error term for the "slow" components and use this to calculate the size of the next step using Eq. (16):

$$H' = H \cdot \left[\frac{\Theta \cdot \in}{l_n}\right]^{1/3} = H \cdot \left[\frac{\Theta}{\text{Error}}\right]^{1/3}$$

STEP 5: If, by taking this step, the end of the large timestep will not be reached, then replace H by H' and return to *STEP 1* for the "slow" components.
Repeat *STEPS 1 to 5* for the "slow" compounds until the size of the next step to be taken will move the integration of the "slow" compounds to beyond the end of the date storage interval.
Truncate the large timestep and use one that will arrive exactly at the end of the interval.
Continue to *Stage 3* of the general algorithm.

STAGE 3: Store the values of all the state variables at the end of the data storage interval. Repeat *Stages 1 to 3* of the general algorithm until one 24 h cycle has been completed.
Check if the differences between the values for all the state variables at the beginning and end of the cycle are less than the stopping criterion.
If this is so, then terminate the integration.
If not, then replace the initial values of the state variables with the most recent values.
Return to *Stage 2* of the general algorithm.
Continue integrating until convergence is achieved.

References

DAHLQUIST, G. and BJORCK, A. (1974) *Numerical methods.* Prentice

Hall Inc. Englewood Cliffs. New Jersey.

GEAR, C.W. (1971) *Numerical initial value problems in ordinary differential equations.* Prentice Hall Inc. Englewood Cliffs. New Jersey.

GEAR, C.W. (1984) Multirate linear multistep methods. *Bit* 24 484-502.

LAMBERT, J.D. (1974) *Computational methods.* John Wiley and Sons. Great Britain.

LAFIDUS, L. and SEINFELD, J.H. (1971) *Numerical solution of ordinary differential equations.* Academic Press. New York.

ORAILOGLU, A. (1983) Software design issues in the implementation of hierarchical display editors. Rept No. UIUCDCS-R-83-1139. Dept. Computer Sci., Univ. of Illinois.