Chapter 18

ESSENTIAL NUMERICAL SUPPORTS FOR KINETIC MODELING SOFTWARE: LINEAR INTEGRATORS

R. C. BOSTON AND T. MCNABB

Clinical Studies, NBC, School of Veterinary Medicine
University of Pennsylvania
Kennett Square, Pennsylvania 19348

P. C. GREIF AND L. A. ZECH

Laboratory of Mathematical Biology
National Institute of Health
Bethesda, Maryland 20892

I. INTRODUCTION

Computers and computer software have had a major impact on the capacity of investigators to probe systems. The power and sophistication of computers have meant that, on the one hand, very complex tasks, such as the population driven analysis of multiple experimental studies (Lyne *et al.*, 1992, Berman and Weiss, 1978) have become possible, and, on the other hand, the effort needed to accomplish tasks, e.g., the manipulation of models and their structural organizations, has been drastically reduced. In developing and refining computer software for a particular (or target) community, care needs to be taken to ensure that the gamut of needs of the community are met. For the modeling community, whose goals seem

primarily linked to understanding and explaining aspects of systems (domains of investigation), this essentially amounts to the provision of a framework to expedite model specification and appraisal. Here specification embraces the incorporation of knowledge, conjecture, information, and data into a schema suitable for computer-supported probing and appraisal involves the evaluation of the conjecture against the information base.

We can identify four classes of support included in modeling software fabricating this framework, viz.,

- Modeling constructs
- Processing support
- Manipulation facilities
- Visual guides.

Modeling constructs are the elements by which models are constructed, and as such they need to have features which render them easily conceptualized as representing entities from the (or any reasonable) domain of investigation. In addition they are of course underpinned by intuitively obvious mathematical structures and require a straight forward set of rules to which they are susceptible in terms of their manipulation. The key is that their functional and abstract senses must be consistent, intuitive, and yet flexible. Examples of (diverse) modeling constructs can be drawn from the SAAM repertoire (see Table I).

Processing support involves, for example, invocation of the machinery underpinning the modeling constructs. Models, as far as the user is concerned, are primarily graphic and lexical entities manipulated in a style which logically advances the investigation. However, to advance the investigation implies access to translation machinery to translate these superficial structures to codified mathematical forms, access to solving machinery to provide requested model predictions, and access to other types of processing machinery as well. The collective responsibility of processing support is to

TABLE I

MODELING CONSTRUCT EXAMPLE FROM SAAM'S "DICTIONARY" (OR MODELING CONSTRUCT LIBRARY)

Construct	Purpose	Conceptual mapping
$L(i,j)$	Identifies rate, mechanism, and transfer sense	Uptake mechanism
$M(i)$	Steady-state pool size	Tissue space
$F(j)$	Differential equation solution	Perturbation response
$QO(j)$	Resetting function	Experimental perturbation

facilitate the integrated analysis of models. The Consam (Boston *et al.*, 1981) modeling software provides the following (examples of) processing support.

• Model	—	DECK	command
• Model solving	—	SOLVe	command
• Parameter refinement	—	ITERate	command
• Batch processing	—	SAAM	command

The manipulation facilities of a modeling environment comprise the collection of resources that expedite user interaction with the model. Here model entry, model modification, and model retention and retrieval are examples of what we have in mind. In addition, provision of support to compare the utility and viability of alternate structural organizations of a model is needed. From the user interface perspective this aspect of modeling software development is extremely challenging on the one hand, easy and efficient access to model manipulation needs to be provided, and, on the other hand, a layer of protection is required to ensure that each modeling step the user takes is not in conflict with either the software rules or past modeling steps.

Finally, critical cues, as visual guides, need to be provided to enable the modeler to appraise progress. For example, the adequacy of data to facilitate identification of a particular parameter or the flexibility of a model to mimic temporal patterns manifest in the data may be issues at the back of the investigators mind as the modeling episode advances. Modeling software must anticipate these and address them in terms that the user is comfortable with as opposed to terms which a scientific programmer might respond to.

In this note we discuss one of the major processing support issues arising in conjunction with the development of modeling software, solving linear systems of differential equations. We start by defining linear systems and describe how they arise in the investigation of biological systems and then move on to relate how they are solved. A key concern of this article is how efficient are the various approaches available for solving linear systems.

II. LINEAR SYSTEMS OF DIFFERENTIAL EQUATIONS

A system of ordinary (as opposed to partial) differential equations is linear if the coefficients of the included derivatives are constant, viz.,

$$\sum_{i=1}^{N} a_i y^{i} = h(t). \tag{1}$$

If N is equal to 1 the system is first order and if $h(t) = 0$, the system is homogeneous. This article will deal with first-order linear homogeneous systems sometimes referred to as initial value problems, viz.,

$$\dot{Y} = LY, \tag{2}$$

where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \end{bmatrix} \dot{Y} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \cdot \\ \cdot \end{bmatrix} Y(0) = \begin{bmatrix} y_1(0) \\ y_2(0) \\ \cdot \\ \cdot \end{bmatrix} \tag{3}$$

and

$$L = \begin{bmatrix} L_{11} & L_{12} & \cdot & \cdot \\ L_{21} & L_{22} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}. \tag{4}$$

The mathematical description of biological processes are rarely representable in terms of systems of the form Eq. (2). The reasons for this are implicit in the form of Eq. (2). First Y in Eq. (2) is self-regulating, yet a biological process would have its own (if any) in-built demand for Y. Second, Y in Eq. (2) transports itself, yet most biological processes have highly refined carriers to meet their needs in conjunction with metabolic deficiencies (see Fig. 1).

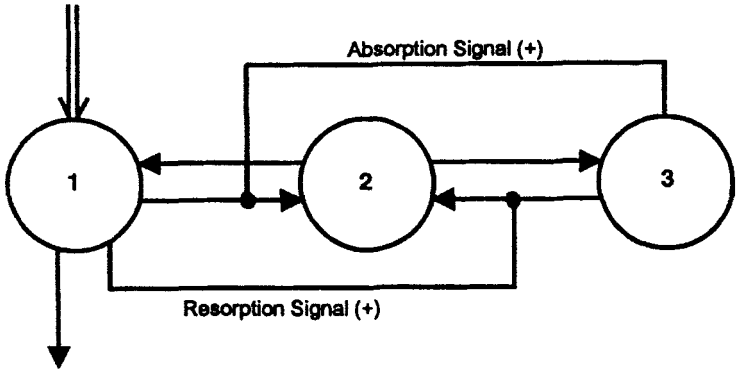

FIG. 1. A typical biological system including metabolic feedback signals.

The question then arises why study first order linear systems (of the form of Eq. (2)) and facilitate their incorporation into modeling software when their relevance seems at best questionable. To answer this we need to note the following.

- The most useful characterization of a biological system may not impose a full dynamic representation of the system but rather a detailed account of the system in a particular metabolic state. For example the characterization of a disease may be most usefully (for example, from the diagnostic perspective) represented in terms of a detailed exposure of its salient aspects at a particular metabolic state of the host
- The response of a nonlinear system at steady state (see above) to a small perturbation is linear, i.e., describable as a set of first order linear differential equations (of the form Eq. (2)). Furthermore, kinetic models represented by such a set of linear differential equations contain many helpful pieces of information concerning the host nonlinear system, viz., the number of exchanging (metabolic) pools, the rate of exchange among the (metabolic) pools, and the size of the (metabolic) pools.

If the nonlinear system represented as

$$Y` = h(Y) + U \tag{5}$$

is perturbed by amount Y^*, then the time profile of Y^* will be given by

$$Y`^* = h` (Y)Y^*, \tag{6}$$

where $h(.)$ is some (nonlinear) function, $h`()$ is the Jacobian of h and U is a steady input to Y.

In Fig. 2A we show that the response of a nonlinear system (of the form Eq. (5)) in steady state to a small perturbation is linear (with parameter predictable from Eq. (6)). In Fig. 2B we demonstrate the effect of non steady state on the response. In Fig. 2C we see the effect of an excessively large perturbation on the response. Further, in Fig. 2D we see the effect of the degree of nonlinearity of the system on the perturbation response.

For small systems involving three or fewer components (or exchanging pools in the instance of kinetic models) the solution to Eq. (2) can be mapped analytically from its exponential form

$$Y = A\exp^{-\alpha t} \tag{7}$$

to its "kinetic" model form

$$L = A\alpha A^{-1}, \tag{8}$$

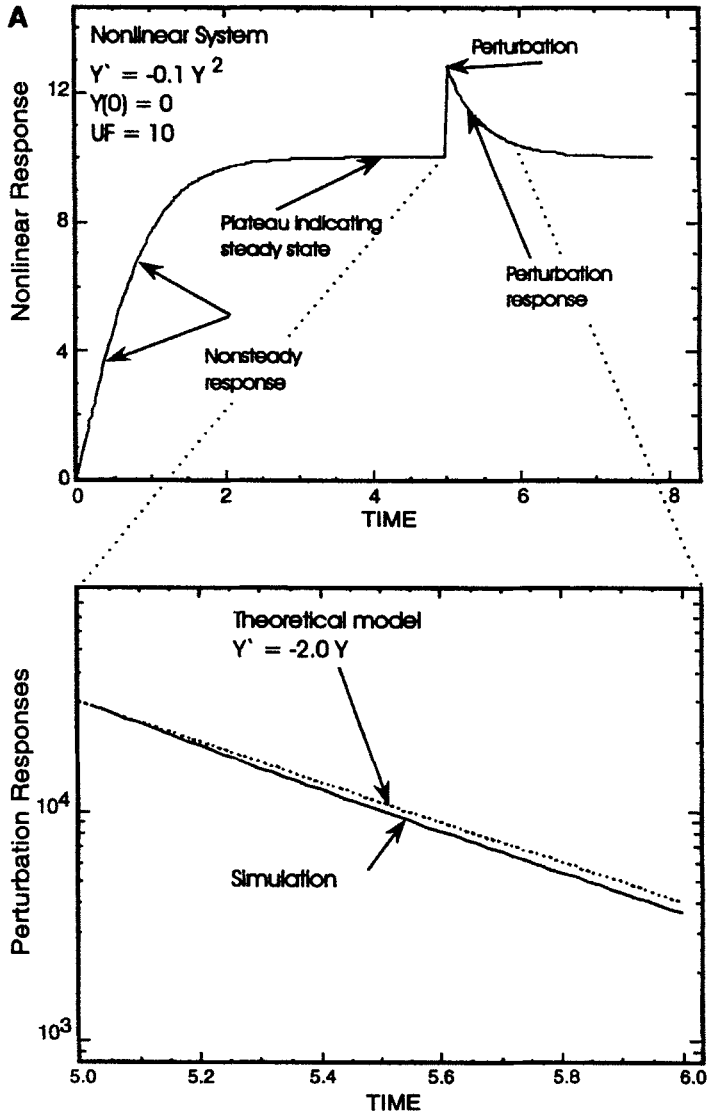

FIG. 2. (A) The response of a nonlinear system at steady state to a small perturbation. The theoretical and simulated responses are presented. (B) The difference in the responses of a nonlinear system to a small perturbation when it is at steady state versus when it is not at steady state. (C) The response of a nonlinear system at steady state to a large perturbation. (D) The effect of the degree of nonlinearity on the response of a nonlinear system at steady state to a small perturbation. Predicted plateaus and slopes are presented in parentheses.

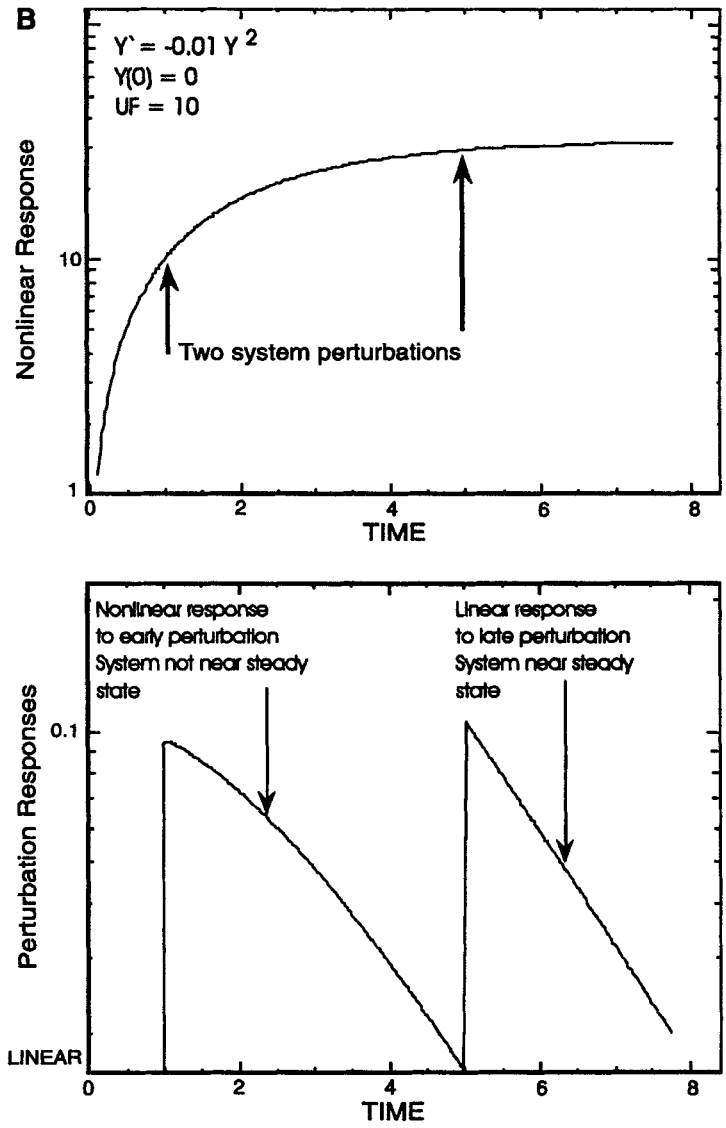

FIG. 2. *Continued*

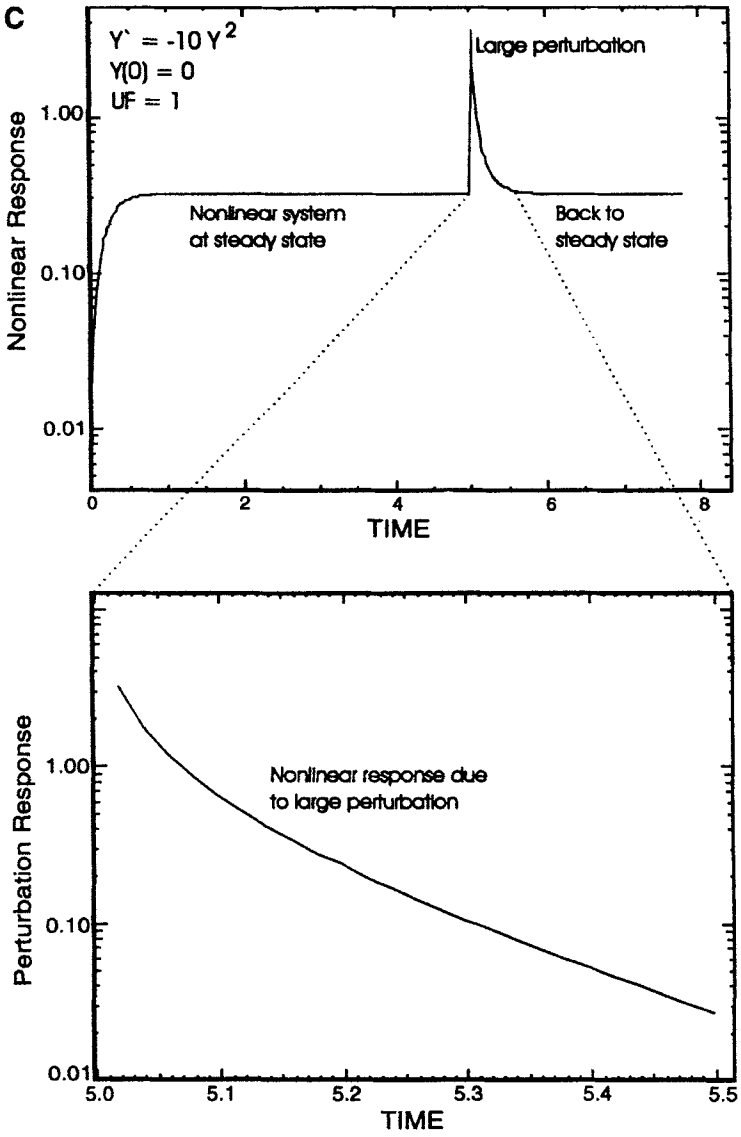

FIG. 2. *Continued*

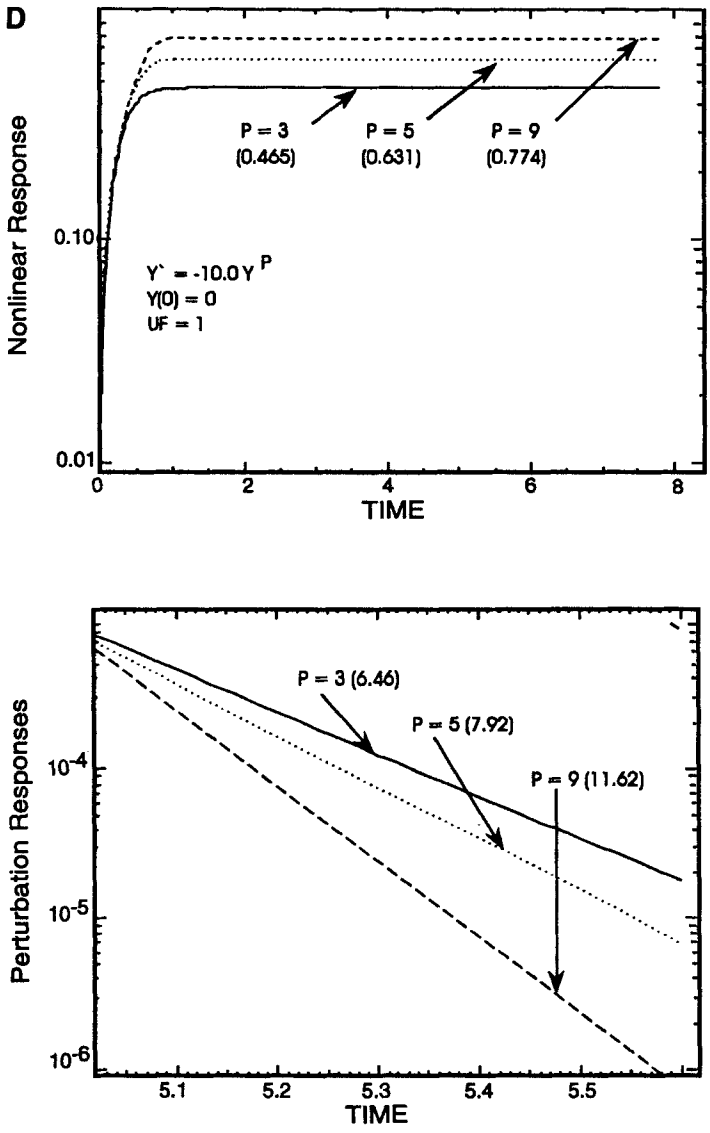

FIG. 2. *Continued*

where, of course, some, or possibly all, of the L_{ij} in Eq. (2) may be negative. It will nevertheless be found most efficient to solve Eq. (2) numerically for the required solution points (time points for which solutions are required) using numerical integrators commonly supported in modeling software.

III. MODELING SOFTWARE AND LINEAR INTEGRATORS

Since we are dealing with first order linear homogeneous systems the computational complexity associated with solving these systems can be considered in terms of the L matrix, though the number of solutions and the integration domain will be secondary issues as well.

There are from our perspective four classes of problems of particular interest in the linear homogeneous set: stiff problems (in which the diagonal elements of the coefficient matrix vary over a large range), problems with oscillating solutions, problems whose coefficient matrix is diagonally dominant (in which the diagonal elements of the coefficient matrix are all considerably larger that the surrounding elements), and other problems. Modeling software not only needs to provide a battery of numerical integrators to address these four classes of problems but also should have the potential to guide the user in terms of matching the integrator used for a problem with the problem characteristics.

For our comparative evaluation of SAAM's Chu Berman (Chu and Berman, 1974) integrator (also referred to as model code 10 in SAAM and CONSAM) we will be considering as alternate integrators the Runge Kutta 4/2 integrator (Press *et al.*, 1987), the Bulirsch Stoer integrator (Press *et al.*, 1987), and Petzold's DASSL integrator (Petzold, 1983).

Our selection of this group of integrators was driven by the following considerations. Runge Kutta (RK) is currently available in the SAAM software, is a tried and proven procedure, and is known to perform at least competitively for nonstiff systems. Bulirsch Stoer (BS) is gaining considerable appeal as a standard, or default, integrator in all but stiff systems with excellent accuracy reports (Acton, 1970, Kahaner *et al.*, 1989). Petzold's DASSL (Petzold) reflects the state-of-the-art in dealing numerically with stiff differential algebraic systems and certainly warrants competitive evaluation against the foregoing. Table II shows the computer language, compiler, and arithmetic precision used in conjunction with generation of executable versions of these integrators.

A. THE CHU BERMAN CONSTANT COEFFICIENT DIFFERENTIAL EQUATION SOLVER

The Chu Berman constant coefficient differential equation solver is a single step integrator which operates on the same principle as the accuracy

TABLE II
COMPILERS USED[a]

Integrator	Language	Compiler	Precision	Size
SAAM 31[b]	Fortran	Watcom	SP	32
CBCCDS[c]	C	Watcom	DP	64
RK[d]	C	Watcom	DP	64
Petzold[e]	Fortran	Watcom	DP	64
XC[f]	C	Borland	Long double	80
BS[g]	Fortran	Watcom	DP	64

[a] Details of compilers used to generate machine images of the code for the four integrators. SP denotes single precision, DP denotes double precision. Size is the number of binary digits used for numerical coding. All compilation and processing was performed on a Gateway 2000 486DX2-50 computer.

[b] The Chu–Berman numerical integrator in the code of SAAM31.

[c] CBCCDS denotes a code C-coded version of the Chu Berman constant coefficient differential solver.

[d] RK denotes the variable step size fourth order Runga Kutta integrator.

[e] Petzold denotes Petzold's DASSL.

[f] XC denotes values obtained from analytical solutions.

[g] BS denotes Bulirsch Stoer integrator.

controlled version of the Runge Kutta integrator (see also Rice, 1983). Figure 3A presents an overview of its step-by-step processing flow and the critical boxes (numbered processing steps in Fig. 3A) are exploded in Figs. 3B and 3C to expose the necessary detail.

The goal of the integrator is to provide acceptably accurate solutions to the differential equations at the user nominated (solution) points (Fig. 4A) with the greatest possible computational efficiency. The process starts with the determination of an integration step (box 2 Fig. 3B) (which may fall short of the required solution point, Fig. 4B) and the estimation of a differential equation solution coinciding with this step (box 3 Fig. 3B). To derive the solution the integrator uses a deconvolution technique in estimating the contribution of the linear term in the Taylor series about the dominant diagonal of the coefficient matrix. We see that the diagonal terms in fact translate to component exponentials in the first stage of deriving the solution. Next the error of the solution is estimated using the quadratic term in the Taylor series expansion (box 4 Fig. 3C). If the error is unacceptably large, the step is halved and the entire process is repeated for this new step (box 5 Fig. 3C and Fig. 4C). If the error is tolerable the step is accepted and a new step proposed.

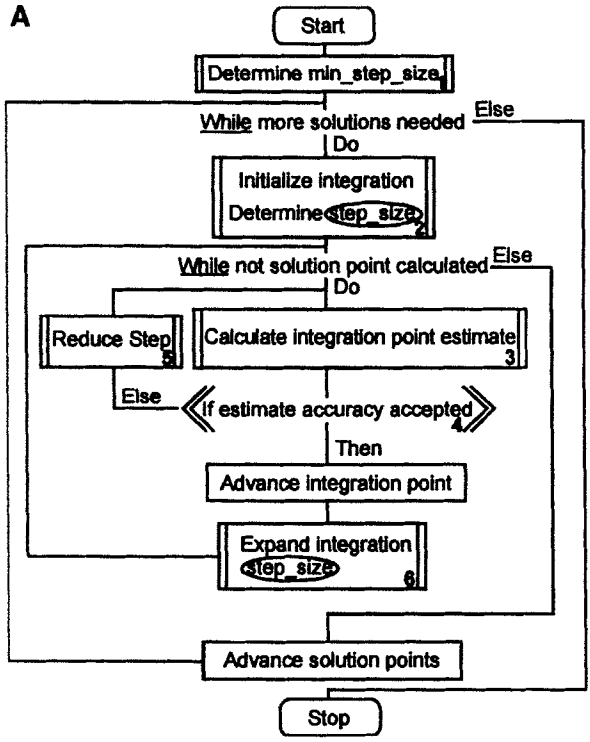

FIG. 3. (A) An overview of the operation of the Chu–Berman constant coefficient solver (CBCCS). (B) Exploded details of boxes 1, 2, and 3 of A showing the determination of the minimum step size, the determination of the actual step size, and the determination of the first de solution estimate for the integration point. (C) Exploded details of boxes 4, 5, and 6 of A showing the solution acceptance process, the step reduction process, and the step expansion process.

The integrator uses a number of intelligent features to improve its performance.

- If a proposed integration step is within 30% of the last step, the step taken is one of the size of the last step (Fig. 4D).
- If the accuracy warrants it, the step is expanded, thus potentially reducing the number of steps per solution point (box 6 Fig. 3C).
- If an integration step would take us to within 30% of a solution point, a step size necessary to hit the solution point is taken (Fig. 4E).
- Decayed components of stiff systems cease to influence step size calculations once their magnitude falls below some acceptably small value.

B **1**
Minimum Step

$$v_I = \left(\sum_{\substack{I=0 \\ \neq J}}^{n} |a_{IJ}| \right)$$

$$MinStep = \frac{StepFactor}{\max_J(v_J)}$$

2
Step Size

$$StepSize(h) = \frac{StepFactor}{\phi}$$

$$\phi = MAX_J \left(MIN_J \left(v_J, \left| \frac{\dot{Q}_J}{Q_J} \right| \right) \right)$$

3
Solution Estimate

$$\alpha_I = \frac{\exp(a_{II}h) - 1}{a_{II}}$$

$$\beta_I = \frac{\exp(a_{II}h) - 1 - a_{II}h}{a_{II}^2}$$

$$x_I^1 = \exp(a_{II}h)x_I^0 + \alpha_I \sum_J^n a_{IJ}x_J^0 + \beta_I \sum_J^n a_{IJ}\dot{x}_J^0$$

FIG. 3. *Continued*

B. EVALUATION OF THE LINEAR INTEGRATORS

1. Test Problem Selection

To examine integrator performance 10 test problems were selected according to the following criteria.

- Used in the literature specifically for testing integrators.
- Comprised problems for which each integrator was potentially well suited as well as problems for which it was likely that they were not well suited.

C **4**
Acceptance Criterion

$$\text{If } X_{max} = MAX_j\left\{\left|\frac{\delta_j h}{x_j^2}\right|\right\} < TolerableRelativeError$$

Then accept step

$$\delta_i = \sum_j a_{ij}\Delta x_j^0$$

$$\Delta x_j^0 = x_j^1 - x_j^0 - h\dot{x}_j^0$$

$$x_j^2 = x_j^1 + \frac{(2\beta_j - h^2)\delta_j}{h^2 a_{jj}}$$

5
Step Reduction

$$\text{Else If } h > MinStep$$

$$\text{Then } h \Leftarrow \frac{h}{2}$$

$$\text{Else } h = MinStep$$

6
Step Expansion

$$h \Leftarrow h\left(\frac{TolerableRelativeError}{X_{max}}\right)^{Gain\ Control}$$

FIG. 3. *Continued*

- Of the form likely to be encountered in practical kinetic modeling episodes.
- Linear or reducible to a linear form.
- Analytic solution available or derivable.
- Modifiable to exhaustively test the integrators to avoid test machine features as factors influencing the results.

Aspects of the test problems are summarized in Table III.

To enable accurate assessment of the speed of the integrators four combinations of solution requests were specified, a *t* series, a *e* series, an *m* series, and a replicated *t* series. The *t* series was a typical irregular set of around 10 solution point requests (in fact each test set included roughly this number of solution requests), similar to a simple modeling run, the *e* series was an exponentially expanding set of solution points similar to what may be present in a data set relating to a tracer experiment, and the *m* series was

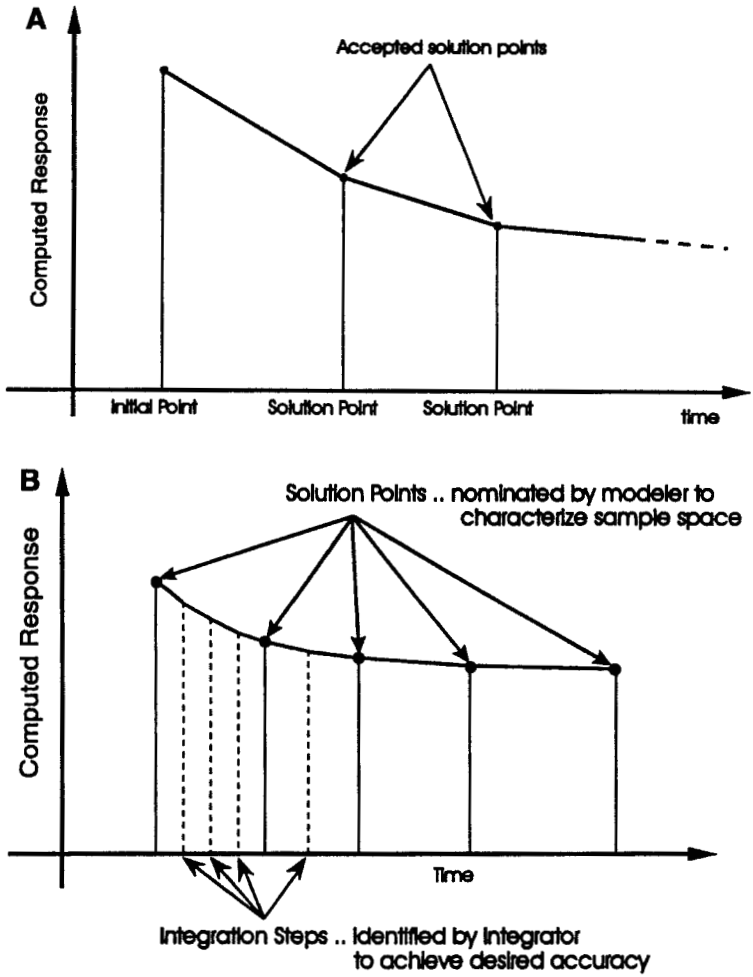

FIG. 4. (A) Solution points negotiated during an integration run. (B) Integration steps negotiated in conjunction with determination of solutions. The first solution point requires four integration steps, the second solution point requires two and the third and fourth solution points require only one integration step each. (C) A demonstration of the step halving process. The initial integration point has an unacceptably large error and so the step must be halved. (D) Improved processing speed is achieved by wherever possible using the calculations performed for the "last" step. (E) When an integration step will advance the solution to within (or beyond) 30% of a required solution point the actual step taken is one that will assure a solution point "hit."

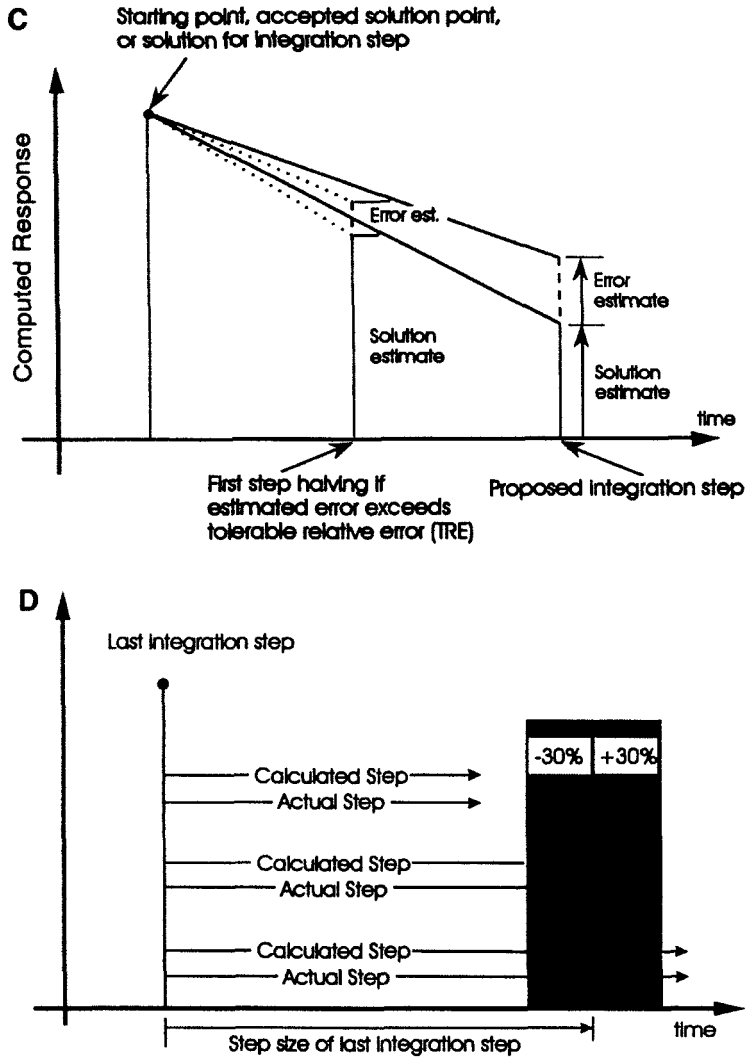

FIG. 4. *Continued*

a uniformly expanding set of points similar to what would be created in conjunction with SAAM's data generation facility. Finally, to allow large enough runs to remove the influence of the computer clock on our speed tests we replicated the t series runs for up to 10,000 repeats. In Table IV we summarize the solution points associated with each of the test sets

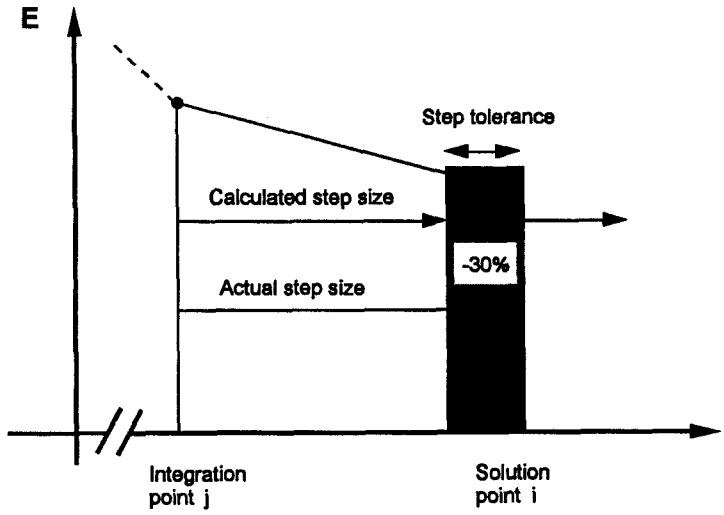

FIG. 4. *Continued*

2. *The Speed of the Integrators*

In Tables VA and VB we summarize the computation time take for each integrator with each problem set. Here we note the following.

- For problems involving 100 solution requests the CBCCDS was consistently faster than the other integrators except for the Petzold integrator when applied to test *t5*.
- For the heaviest test involving some 10,000 solution requests the superiority of the Petzold integrator in dealing with very stiff systems emerged.
- For its preferred domain of operation the CBCCDS was as fast as Petzold's integrator, 13 times faster that the RK integrator and 5 times faster than the BS integrator.
- For small problem sets (*e* and *m* series) representative of small kinetic modeling runs CBCCDS, RK, and BS had substantial speed advantages over the Petzold scheme. CBCCDS was slightly superior to the other two.

TABLE III

DETAILS OF THE TEST PROBLEMS

ID	Source	Math		Analytic
*t*1	Lapidus and Seinfeld (1971, p. 84. I)	$y_1' = -y_1$	$y_1(0) = 1.0$	$y_1(x) = e^{-x}$
*t*2	Lapidus and Seinfeld (1971, p. 84. II)	$y_1' = y_1$	$y_1(0) = 1.0$	$y_1(x) = e^{x}$
*t*3	Lapidus and Seinfeld (1971, p. 84. III)	$y_1' = y_2 + y_1$	$y(0) = 0.0$	$y_1(x) = e^{x} - 1$
		$y_2' = 0.0$	$y_2(0) = 1.0$	$y_2(x) = 0.0$
*t*5	Lapidus and Seinfeld (1971, p. 84. VI)	$y_1' = -49.9y_2 - 0.1y_1$	$y_1(0) = 2.0$	$y_1(x) = e^{-0.1x} + e^{-50x}$
		$y_2' = -50y_2$	$y_2(0) = 1.0$	$y_2(x) = e^{-50x}$
		$y_3' = 70y_2 - 120y_3$	$y_3(0) = 2.0$	$y_3(x) = e^{-50x} + e^{-120x}$
*t*13	Chemical kinetics	$y_1' = y_2 - y_1$	$y_1(0) = 2.0$	$y(x) = 1 + \frac{e^{-3x}}{2} + \frac{e^{-x}}{2}$
		$y_2' = y_1 - 2y_2 + y_3$	$y_2(0) = 0.0$	$y_2(x) = 1 - e^{-3x}$
		$y_3' = y_2 - y_3$	$y_3(0) = 1.0$	$y_3(x) = 1 + \frac{e^{-3x}}{2} - \frac{e^{-x}}{2}$
*t*14	Test 05, problem 14	$y_1' = -y_1$	$y_1(0) = 1.0$	$y_1(x) = e^{-x}$
		$y_2' = -y_2$	$y(0) = -1.0$	$y_2(x) = -e^{-x}$
*t*20	Braun (1986, p. 379)	$y_1' = -y_1$	$y(0) = 0.0$	$y_1(x) = 0.0$
		$y_2' = -2y_1 + 2y_3 - y_2$	$y_2(0) = 0.0$	$y_2(x) = 2e^{-x} \sin x \cos x$
		$y_3' = -3y_1 - 2y_2 - y_3$	$y_3(0) = 1.0$	$y_3(x) = 2e^{-x} \cos x^2 - e^{-x}$
*t*21	Braun (1986, p. 381)	$y_1' = 2y_1 - 3y_2$	$y(0) = 0.0$	$y(x) = \frac{6}{7}x - \frac{6}{49} + \frac{6}{49}e^{-7x}$
		$y_2' = -6y_2 - 2y_3$	$y_2(0) = 0.0$	$y_2(x) = \frac{4}{7}x - \frac{18}{49} + \frac{18}{49}e^{-7x}$
		$y_3' = -6y_1 - 3y_3$	$y_3(0) = 1.0$	$y_3(x) = -\frac{12}{7}x + \frac{40}{49} + \frac{9}{49}e^{-7x}$
*t*30	Lapidus and Seinfeld (1971, p. 84. V)	$y_1' = y_2$	$y_1(0) = 1.0$	$y_1(x) = \sin x + \cos x$
		$y_2' = -y_1$	$y_2(0) = 1.0$	$y_2(x) = \cos x - \sin x$
*t*77	Chu and Berman (1974)	$y_1' = -200y_1 - 1991y_2 + 2000y_3 - 199y_4$	$y_1(0) = 10.0$	$y_1(x) = 10 + 10e^{-200x} - xe^{-x} - 10e^{-x}$
		$y_2' = -y_2$	$y_2(0) = 1.0$	$y_2(x) = e^{-x}$
		$y_3' = 0.0$	$y_3(0) = 1.0$	$y_3(x) = 1.0$
		$y_4' = y_2 - y_4$	$y_4(0) = 0.0$	$y_4(x) = xe^{-x}$

TABLE IV

SOLUTION POINTS USED FOR EACH TEST PROBLEM AND SERIES

Problem	Number of compartments evaluated	Normal points (t series)
$t1$	1	0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0
$t2$	1	0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0
$t3$	2	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0
$t5$	3	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0, 20.0, 40.0
$t7$	4	0.001, 0.005, 0.01, 0.028, 0.05, 0.1, 0.2, 0.5, 0.7, 1.0, 2.0, 10.0
$t13$	3	0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 1.0, 2.0, 3.0, 5.0, 7.0
$t14$	2	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0, 2.5
$t20$	3	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0, 2.5
$t21$	3	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0, 2.5
$t30$	2	0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 4.0, 6.0, 8.0, 10.0
		Exponential solution (e series)
All	All	1, 2, 4, 8, 16, 32, 64
	Number of compartments evaluated	Incremental solution points (m series)
All	All	10, 20, 30, 40, 50, 60, 70, 80, 90, 100

3. *Integrator Precision*

In Table VI we summarize the accuracy of each integrator for each test problem. Here accuracy is defined to be the lowest number of digits of agreement between the exact solution and the particular integrator's results for each of the component responses and considering each solution point. From Table VI we note the following.

- For all tests the CBCCDS integrator gave at least the accuracy its settings suggested.
- For its preferred problem set the CBCCDS integrator was convincingly more accurate that the others.
- The Petzold and BS integrators each failed on a number of problems.
- The RK integrator performed with reasonable consistency on all problems.

TABLE VA

PROCESSING TIME IN SECONDS FOR EACH INTEGRATOR AND TEST PROBLEM, USING THE *e* AND *m* SERIES OF SOLUTION POINTS

Problem ID:	CBCCS C/Watcom	CBCCS 100 C/Watcom	RK C/Watcom	Petzold F/Watcom	BS F/Watcom	SAAM 31 F/Watcom
*e*1	0	0.05	0.06	0.22	0.05	0.109
*e*2	0.05	0.11	0.06	0.16	0.05	0.059
*e*3	0	0.05	0.11	0.22	0.05	0.059
*e*5	0.11	8.6	0.17	0.17	1.92	0.109
*e*77	0.05	4.7	1.5	0.49	2.86	11.758
*e*13	0	0.88	0	0.22	0.06	0.168
*e*14	0.06	0.11	0.05	0.33	0.11	0.109
*e*20	0.17	1.4	0.38	0.66	0.22	1.148
*e*21	0.06	2.6	0.06	0.22	0.22	0.281
*e*30	0.05	6.1	0.16	0.33	0.06	0.820
		CBCCS C/Watcom				
*m*1	0.06	0.28	0.05	0.33	0.05	1.100
*m*2	0.0	0.22	0.11	0.38	0.11	1.369
*m*3	0.0	0.44	0.16	0.72	0.11	1.480
*m*5	0.11	20.00	0.27	0.11	2.96	1.590
*m*77	0.11	11.00	0.06	0.55	5.05	4.117
*m*13	0.0	2.9	0.06	0.16	0.11	2.359
*m*14	0.0	0.33	0.22	0.50	0.11	1.590
*m*20	0.16	33.00	0.11	1.04	0.28	2.359
*m*21	0.06	9.2	0.22	0.17	0.33	2.199
*m*30	0.11	23.00	2.3	0.88	0.11	2.371

• Even for tests for which the CBCCDS integrator was unsuited it returned solutions of acceptable accuracy.

IV. CONCLUSION

From this work we may draw the following conclusions.

• With the exception of two instances with problem *t*20, the CBCCDS integrator delivered accuracy of better than 1% irrespective of the nature of the linear problem.

• Systems of linear differential equations for which the stiffness is greater than 1000 or those for which the solution involves periodic functions present the greatest problems for the CBCCDS integrator.

TABLE VB

PROCESSING TIME IN SECONDS FOR EACH INTEGRATOR AND TEST PROBLEM, FOR MULTIPLE RUNS OF THE *t*-SERIES SOLUTION POINTS

Integrator:	CBCCS				Petzold				RK		BS	
Repeat:	10	100	1000	10000	10	100	1000	10000	1000	10000	1000	10000
*t*1	0	0.11	0.83	7.9	0.44	0.44	0.93	5.33	11	110	3.9	38.78
*t*2	0	0.05	0.77	7.7	0.39	0.38	0.88	5.38	11	110	3.57	36.03
*t*3	0	0.16	1.4	13	0.5	0.6	1.32	8.18	19	190	6.32	61.9
*t*5	0.88	8.4	84	840	0.33	4.9	1.43	10.82	130	1300	1173.7	11737.25
*t*77	0.22	1.8	18	180	0.49	0.66	1.65	11.21	250	2500	491.09	4910.34
*t*13	0.05	0.28	2.7	27	0.44	0.55	1.32	8.79	18	180	12.36	123.58
*t*14	0	0.17	1.2	11	0.17	0.22	0.82	6.37	6.6	66	4.01	39.93
*t*20	0.05	0.33	3	30	0.43	0.49	1.2	8.24	21	210	9.28	92.61
*t*21	0.06	0.5	4.8	48	0.55	0.55	1.21	8.07	22	220	54.70	547
*t*30	0.11	0.93	9.2	n/a	0.6	0.66	n/a	n/a	21	n/a	5.88	n/a

TABLE VI

INTEGRATOR ACCURACY[a]

	CBCCS				RK				Petzold				BS			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
*t*1	16/15				16/7				12/9				15/9			
*t*2	16/16				15/7				12/9				16/9			
*t*3	16/14	16/16			16/7	16/16			13/7	16/16			15/7	16/16		
*t*5	7/4	16/16	16/4		15/7	16/6	16/6		9/7	8/0	9/0		9/7	12/0	12/7	
*t*77	11/6	16/15	16/16	11/3	10/7	16/10	16/16	16/9	12/7	11/8	16/16	10/7	14/7	14/8	16/16	14/8
*t*13	9/4	7/4	9/4		13/8	11/8	13/8		13/8	15/8	13/8		16/8	16/8	16/8	
*t*14	16/16	16/16			16/7	16/7			12/9	12/9			16/9	16/9		
*t*20	16/16	10/2	12/2		16/16	16/7	16/7		16/16	13/7	13/9		16/16	14/8	13/9	
*t*21	7/4	8/4	9/3		9/6	9/7	10/9		10/6	10/7	11/8		3/0	3/0	3/0	
*t*30	10/3	9/3			16/7	16/7			13/9	12/9			16/9	16/9		

[a] The accuracy of each integrator for each test problem. Here accuracy is defined as the worst number of digits of agreement or any solution within each test problem run when comparing the particular integrator results with the "exact" results. Each compartment accuracy is indicated by its ordinal assignment of digits. The slash (/) separates compartment precision.

- The CBCCDS integrator was the fastest of all integrators for problems of small to medium size.
- Although a consistent performer the slowness of the Runge Kutta integrator rule it out in favor of the Petzold integrator in those areas where an alternative to the CBCCDS integrator is needed.
- The Bulirsch Stoer integrator performed with consistent accuracy but again was too slow to compete with Petzold's scheme as an alternative to the CBCCDS integrator.
- The CBCCDS integrator is justifiably placed in SAAM as the default integrator.

REFERENCES

Acton, F. S. (1970). "Numerical Methods that Work. Harper & Row, New York.

Berman, M., and Weiss, M. F. (1978). "SAAM Manual," DHEW Publ. No. (NIH) 78-180. U.S. Govt. Printing Office Washington, DC.

Boston, R. C., Greif, P. C., and Berman, M. (1981). Conversational SAAM: An interactive program for the kinetic analysis of biological systems. *Comput. Methods Programs Biomed.* **13,** 111–119.

Braun, M. (1986). "Differential Equations and Their Applications." Springer-Verlag, New York.

Chu, S. C., and Berman, M. (1974). An exponential method for the solution of systems of ordinary differential equations. *Commun. ACM* **17,** 699–702.

Frost, A. A., and Peason, R. G. (1961). "Kinetics and Mechanism." Wiley, New York.

Kahaner, D., Moler, C., and Nash, S. (1989). "Numerical Methods and Software." Prentice Hall, Englewood Cliffs, NJ.

Lapidus, L., and Seinfeld, J. H. (1971). "Numerical Solution of Ordinary Differential Equations." Academic Press, New York.

Lyne, A., Boston, R., Pettigrew, K., and Zech, L. (1992). EMSA: A SAAM service for the estimation of population parameters based on model fits to identically replicated experiments. *Comput. Methods Programs Biomed.* **38,** 117–151.

Petzold, L. (1983). A description of DASSL: A differential/algebraic system solver. *In* "Scientific Computing" (R. Stepleman *et al.,* eds.), pp. 65–68. IMACS/North-Holland Printing Co., Amsterdam.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vettling, W. T. (1987). "Numerical Recipes: The Art of Scientific Computing." Cambridge Univ. Press, New York.

Rice, J. R. (1983). "Numerical Methods, Software, and Analysis: IMSL Reference Edition." McGraw Hill, New York.