Biochem. J. (1977) **165**, 55–59
Printed in Great Britain

An Automatic Method for Deriving Steady-State Rate Equations

By ATHEL CORNISH-BOWDEN
Department of Biochemistry, University of Birmingham, P.O. Box 363, *Birmingham B15 2TT, U.K.*

(*Received* 4 *October* 1976)

A method is described for systematically deriving steady-state rate equations. It is based on the schematic method of King & Altman [*J. Phys. Chem.* (1956) **60**, 1375–1378], but is expressed in purely algebraic terms. It is suitable for implementation as a computer program, and a program has been written in FORTRAN IV and deposited as Supplementary Publication SUP 50078 (12 pages) at the British Library (Lending Division), Boston Spa, Wetherby, West Yorkshire LS23 7BQ, U.K., from whom copies can be obtained on the terms indicated in *Biochem. J.* (1977) **161**, 1–2.

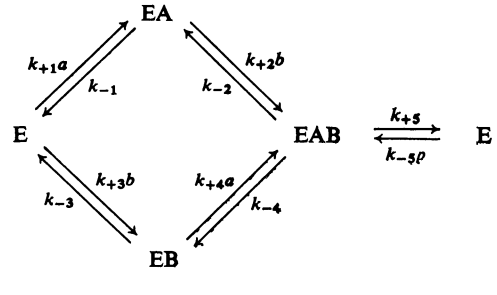
In principle, the steady-state rate equation for any enzyme-catalysed reaction can be derived by symbolic solution of the simultaneous equations that result from combining the conservation equation with the steady-state expressions for all of the intermediates. This method was successfully applied by such workers as Botts & Morales (1953) to some surprisingly complex mechanisms, but it is very tedious for any but trivially simple mechanisms, and it was not until the publication of the schematic method of King & Altman (1956) that rapid development of enzyme kinetics was practicable. The modifications introduced by Volkenstein & Goldstein (1966) and Cha (1968) added substantially to the power of the King–Altman method, and some alternative methods are now available, such as those described by Fromm (1970), Orsi (1972), Seshagiri (1972), Ainsworth (1974) and Indge & Childs (1976). Nonetheless, there remain some obstacles to the practical application of the steady-state assumption, which have become more serious in recent years with increasing recognition that there may be numerous enzymes that do not obey simple mechanisms (see, e.g., Meunier *et al.*, 1974; Childs & Bardsley, 1975; Crabbe & Bardsley, 1976; Storer & Cornish-Bowden, 1977). These stem from the fact that a complex mechanism usually requires a complex rate equation, and any manual method of deriving it is likely to be tedious and prone to human error. So there appears to be a need for a fully automatic version of the King–Altman method that can be expressed as a computer program. The present paper describes such a method, which has been extensively used in our laboratory in the study of glucokinase from rat liver (Storer & Cornish-Bowden, 1977).

Principle

The King–Altman method is, in effect, a set of geometric rules designed to simplify an algebraic procedure. The geometric aspects are irrelevant to the computational problem, however, and it is instructive to reformulate the method in purely algebraic terms. Any mechanism, for example the one shown in Scheme 1, is defined by an array of n enzyme forms and an array of up to $n(n-1)$ first-order (or pseudo-first-order) rate constants for the reactions between them, as shown in Table 1.

In the steady state the proportion of any enzyme form E_m can be written as $[E_m]/e_0 = N_m/D$, where e_0 is the total enzyme concentration, N_m is the specific numerator expression for E_m, and $D = \sum N_m$ is the sum of all such numerator expressions. The King–Altman method provides a way of expressing each numerator N_m as the sum of a series of products of rate constants that satisfy the following rules.

(1) Each product contains $(n-1)$ rate constants. (2) There are no rate constants for reactions leading directly from E_m, i.e. no rate constants can be taken from the mth row of rate constants arranged as in Table 1. (3) There is one rate constant only for a reaction leading directly away from each enzyme form apart from E_m, i.e. there is one rate constant from every row except the mth of the rate constants arranged as in Table 1. (4) All rate constants refer to reactions that exist in the mechanism, e.g. in Table 1 there is no rate constant in the fourth column of the



Scheme 1. *Random-order ternary-complex mechanism*

Table 1. *Array of enzyme forms and rate constants for Scheme* 1

The Table shows the rate constants for all of the reactions connecting pairs of enzyme forms in the mechanism shown in Scheme 1. All rate constants are expressed in first-order or pseudo-first-order form; for example, the second-order rate constant k_{+1} is multiplied by the substrate concentration a to give the pseudo-first-order rate constant $k_{+1}a$. Reactions that are missing from the mechanism, such as the direct interconversion of EA and EB, are assigned zero rate constants.

From \ To	... E	EA	EAB	EB
E	—	$k_{+1}a$	$k_{-5}p$	$k_{+3}b$
EA	k_{-1}	—	$k_{+2}b$	0
EAB	k_{+5}	k_{-2}	—	k_{-4}
EB	k_{-3}	0	$k_{+4}a$	—

second row, because there is no reaction in Scheme 1 from EA to EB. (5) There is at least one rate constant for a reaction leading directly to E_m, i.e. there is at least one rate constant from the mth column. (6) No cyclic reactions can be represented in the product, e.g. the product $k_{-5}p \cdot k_{-1} \cdot k_{-2}$ from Table 1 represents the cyclic reaction $E \rightarrow EAB \rightarrow EA \rightarrow E$, and is therefore forbidden.

For an explanation of the reasons for these rules, see King & Altman (1956) or Cornish-Bowden (1976*a*). They are expressed somewhat redundantly to avoid ambiguity; it is, for example, impossible to satisfy rules 2 and 3 without also satisfying rule 1, and it is impossible to satisfy rules 2 and 6 without also satisfying rule 5. Thus in principle it is unnecessary to ensure that (for example) rules 1 and 5 are satisfied, though there may be practical advantages in doing so, as I shall discuss below.

It is not difficult to generate in turn all possible products that satisfy rules 1–4; so in principle N_m can be evaluated by checking each product as it is generated and including it in the summation only if it satisfies rules 5 and 6. It is not necessary to repeat the procedure for each enzyme form, i.e. for each value of m, because any product in the summation for N_1 can be transposed immediately into the corresponding product in the summations for N_2, N_3 etc. Thus it is sufficient to generate and test a series of products for N_1.

To generate such a series, the first row of the table of rate constants is ignored (to satisfy rule 2), and the first product is obtained by taking the first non-zero rate constant from every row except the first; for the example shown in Table 1, this gives $k_{-1} \cdot k_{+5} \cdot k_{-3}$. This is converted into the second product by replacing the last rate constant with the second non-zero rate constant of the nth row, giving $k_{-1} \cdot k_{+5} \cdot k_{+4}a$. In the case of Table 1 this exhausts the nth row, and the next product is obtained by returning to the beginning of the nth row and proceeding to the second non-zero rate constant of the $(n-1)$th row. This process is continued until all possibilities are exhausted, rejecting any products that do not contain at least one rate constant from the first column (to satisfy rule 5). The complete series generated in this way from Table 1 consists of $k_{-1} \cdot k_{+5} \cdot k_{-3}$, $k_{-1} \cdot k_{+5} \cdot k_{+4}a$, $k_{-1} \cdot k_{-2} \cdot k_{-3}$, $k_{-1} \cdot k_{-2} \cdot k_{+4}a$, $k_{-1} \cdot k_{-4} \cdot k_{-3}$, $k_{-1} \cdot k_{-4} \cdot k_{+4}a$, $k_{+2}b \cdot k_{+5} \cdot k_{-3}$, $k_{+2}b \cdot k_{+5} \cdot k_{+4}a$, $k_{+2}b \cdot k_{-2} \cdot k_{-3}$, $k_{+2}b \cdot k_{-4} \cdot k_{-3}$. Because each product is obtained from the previous one by replacing one or more rate constants from the same rows, it follows that rules 1–3 are necessarily obeyed. Further, omission of all zero rate constants ensures that rule 4 is obeyed. This leaves only rule 6 to be tested. As this is much more laborious than testing for rule 5, it is worth while rejecting products that do not satisfy rule 5 before checking rule 6, even though rule 5 is logically redundant.

Cycles are tested (rule 6) by taking each rate constant in a product in turn and following the specified sequence of reactions until the target enzyme form, E_1, is reached. The reaction specified by any rate constant is determined by its column and row in the Table: for example, $k_{+4}a$ occurs in column 3 of row 4 in Table 1, and so it refers to the reaction from E_4 (EB) to E_3 (EAB). So one can identify the next step in any sequence of reactions by finding the rate constant with the same row number as the column number of its precursor. No ambiguity is possible because no product contains more than one rate constant from any row (rule 3). In the product $k_{+2}b \cdot k_{+5} \cdot k_{-3}$ the first rate constant $k_{+2}b$ is from the third column of Table 1, and therefore leads to whatever rate constant is from the third row, i.e. k_{+5}. This terminates the sequence because it is from the first column and so leads directly to E_1. Any rate constant that appears in a tested sequence, i.e. k_{+5} in this example, does not need to be tested again. But k_{-3} did not appear in the test for $k_{+2}b$ and must be tested. It is in column 1 and so leads directly to E_1. Thus the product as a whole contains no cycle and is acceptable. But consider now the product $k_{+2}b \cdot k_{-2} \cdot k_{-3}$: the test for $k_{+2}b$ generates the sequence $E_2 \rightarrow E_3 \rightarrow E_2 \rightarrow E_3 \rightarrow \ldots$ and never leads to E_1. A simple cycle of two reactions is easily recognized, but in general the problem is not trivial, and the simplest solution is to count steps until the $(n-2)$th; if a sequence of $(n-2)$ steps does not end with E_1 it must be cyclic. [It might seem necessary to count $(n-1)$ steps rather than $(n-2)$, but provided rule 5 is satisfied a cycle of more than $(n-2)$ steps is impossible.] This method may seem cumbersome, but a computer cannot 'remember' what enzyme forms have been encountered in a sequence except by compiling a complete list and checking each new enzyme form against every member of the list. It is

more efficient to avoid such repetitive checking by counting as described.

In the example given, eight products are left after rejection of two cyclic products, $k_{-1}\cdot k_{-4}\cdot k_{+4}a$ and $k_{+2}b\cdot k_{-2}\cdot k_{-3}$. N_1 is the sum of these eight, i.e.:

$$\begin{aligned}N_1 = {} & k_{-1}\cdot k_{+5}\cdot k_{-3}+k_{-1}\cdot k_{+5}\cdot k_{+4}a+k_{-1}\cdot k_{-2}\cdot k_{-3}\\ & + k_{-1}\cdot k_{-2}\cdot k_{+4}a + k_{-1}\cdot k_{-4}\cdot k_{-3}\\ & + k_{+2}b\cdot k_{+5}\cdot k_{-3} + k_{+2}b\cdot k_{+5}\cdot k_{+4}a\\ & + k_{+2}b\cdot k_{-4}\cdot k_{-3}\end{aligned}$$

Each product in this sum can be transposed into the corresponding product in N_2 by identifying the pathway from E_2 to E_1 and then reversing the rows and columns of all rate constants along the route. In $k_{-1}\cdot k_{+5}\cdot k_{-3}$ the pathway from E_2 to E_1 is represented by the single rate constant k_{-1}, which is from column 1 and row 2 of Table 1. In N_2 it must be replaced with the rate constant from column 2 and row 1, i.e. $k_{+1}a$, so the whole product becomes $k_{+1}a\cdot k_{+5}\cdot k_{-3}$. In $k_{+2}b\cdot k_{+5}\cdot k_{-3}$ the pathway from E_2 to E_1 is represented by two rate constants, $k_{+2}b\cdot k_{+5}$, and so the product transposed for N_2 is $k_{-2}\cdot k_{-5}p\cdot k_{-3}$. The remaining products in N_2 and the other numerator expressions are obtained similarly.

Repeated rate constants

When carrying out the above procedure it is essential to treat the rate constants for the different steps as different, even if they happen to have the same symbol (see Cornish-Bowden, 1976*b*). In other words each rate constant must be defined by its place in the table of reactions and not by its symbol. For example, if one wished to study the mechanism shown in Scheme 1 with the assumption that the two substrates bound to the enzyme without any interaction, i.e. $k_{+4}=k_{+1}$, $k_{-4}=k_{-1}$, $k_{+3}=k_{+2}$, $k_{-3}=k_{-2}$, the table might be constructed so that $k_{+1}a$, k_{-1}, $k_{+2}b$ and k_{-2} each occurred twice. But in deriving the rate equation it would then be important to treat the two rate constants labelled, for example, $k_{+1}a$, as two different rate constants distinguished by their different positions in the table. In the computer program described below this problem is taken care of automatically and no special precautions on the part of the user are necessary to avoid it.

Irreversible steps

Irreversible steps can be accommodated most efficiently by suppressing the printing of products that contain zero elements. But if any such products occur in the definition of N_1 they must nonetheless be remembered because they may well become non-zero when they are transposed into the definitions of the other numerators. Thus a distinction must be made between missing reactions (e.g. between EA and EB in Scheme 1), which have zero rate constants in both directions, and irreversible reactions (e.g. from EAB to E in Scheme 1 in the special case of the initial rate in the absence of product, i.e. when $p=0$), which have zero rate constants in one direction only. The latter must be treated like any other reactions except in the final printing of results. The computer program described below is written so that any reaction may be included in the mechanism but assigned a zero rate constant in one direction.

Parallel steps

In some mechanisms there may exist two or more parallel steps between a pair of enzyme forms. (For example, in the two-step Michaelis–Menten mechanism the enzyme–substrate complex can be converted into free enzyme either by release of substrate or by release of product.) In such cases the rate constants for the parallel steps must be added together at the outset, i.e. when the table of rate constants is drawn up. Although this is merely a useful but inessential simplification of the schematic method (Volkenstein & Goldstein, 1966), it is an essential part of the algebraic method.

Steps at equilibrium

If some steps in a mechanism are treated as equilibria, the rate equation and its derivation are greatly simplified (Cha, 1968). Each group of enzyme forms at equilibrium with one another is treated as a single form, and all rate constants leading away from the composite form are weighted according to the reactive fraction of the equilibrium. For example, suppose one requires a rate equation for Scheme 1 in which the binding of B to E is treated as an equilibrium. In this case $[EB]/[E]=k_{+3}b/k_{-3}$, and (E+EB) can be treated as a composite species. The fraction of (E+EB) that is capable of binding A to give EA is $1/(1+k_{+3}b/k_{-3})$, and so the rate constant $k_{+1}a$ for conversion of E into EA must be replaced by the weighted rate constant $k_{+1}a/(1+k_{+3}b/k_{-3})$ for conversion of (E+EB) into EA. Applying this approach to all rate constants for reactions that lead away from E or EB one can convert Table 1 into Table 2, which can then be analysed in the same way as before. Although some of the rate constants in Table 2 have a more complicated appearance than the corresponding ones in Table 1, the analysis is much simpler, because there are only three products in each summation instead of eight.

Implementation

The method described has been implemented as a program in FORTRAN IV, which has been extensively used over a 6-year period on four different

Table 2. *Treatment of steps at equilibrium*
This Table is similar to Table 1, but is obtained from Scheme 1 by assuming E and EB to be in equilibrium, so that $[EB]/[E] = k_{+3}b/k_{-3}$, and treating (E+EB) as a composite enzyme form.

From To ...	(E+EB)	EA	EAB
(E+EB)	—	$\frac{k_{+1}k_{-3}a}{k_{-3}+k_{+3}b}$	$\frac{k_{-3}k_{-5}p+k_{+3}k_{+4}ab}{k_{-3}+k_{+3}b}$
EA	k_{-1}	—	$k_{+2}b$
EAB	$k_{-4}+k_{+5}$	k_{-2}	—

computers. The current version is written for an International Computers Ltd. (ICL) 1906A computer and contains some unavoidable but minor machine-dependent features. These are unlikely to present any serious obstacles to running the program on other computers, however, as only slightly different versions were required for use on a Control Data Corporation (CDC) 6400, an English Electric KDF9 and an International Business Machines (IBM) 370. The current version, together with sample input, output, instructions for use and suggestions of modifications likely to be necessary with other computers, has been deposited as Supplementary Publication SUP 50078 at the British Library (Lending Division) Boston Spa, Wetherby, West Yorkshire LS23 7BQ, U.K.

The program accepts alphanumeric names for rate constants and enzyme forms, and prints them in the same way. For example, a rate constant $k_{+1}a$ can be read in and printed as K+1A. Consequently the input is very simple and little or no translation of the output is necessary to make it comprehensible.

The program is written to accept mechanisms of up to ten enzyme forms and provides a complete listing of all numerator terms, provided that there are no more than $1500/(n-1)$ products in each summation, where n is the number of enzyme forms [i.e. not more than $1500/(n-1)$ King–Altman patterns]. If this limit is exceeded, a complete listing of N_1 is given, but the other numerators are truncated after $1500/(n-1)$ products and warnings are printed stating the number of products omitted. The limits are included to avoid wasting core store for simple mechanisms and can readily be increased if necessary.

The program has been tested with many simple mechanisms and some complex ones, such as the general two-substrate–two-product mechanism of Lam & Priest (1972), which includes eight forms and 12 reactions between them. With this mechanism, the program generated 288 products for each numerator, which were compared with the 288 King–Altman patterns listed by Lam & Priest (1972). Apart from the sequence in which they were generated the two lists were identical.

The time required for execution of the program varies somewhat with the computer used, but with modern computers (CDC 6400, ICL 1906A or IBM 370) the central processor time has never exceeded a few seconds. For simple mechanisms it is always much less than 1 s, and, for the mechanism of Lam & Priest (1972) mentioned above, the program required 8.9 s for execution on an IBM 370 computer. The amount of core store required also varies with different computers, but is typically about one-half of the minimum required by the FORTRAN compiler; in other words, the program is easily small enough to run on any machine that can accommodate a FORTRAN compiler.

Discussion

The protocol described in the present paper was developed as a stage in the writing of a computer program, but there is no reason why it should not be used manually as an alternative to other methods of deriving rate equations. As all valid methods must lead to equivalent results it is a matter of personal preference which should be used and there can be no definitive assessment of which is 'best'. But it may be helpful to outline the special advantages of the currently available methods.

For complex mechanisms, the main advantage of the original schematic method of King & Altman (1956) is that it is immediately obvious to the eye which patterns are valid and which are not, and so no tests are necessary. But one must set against this the difficulty of being certain that all possible patterns have been found. This difficulty is decreased by following the recommendations of Volkenstein & Goldstein (1966); but then there is the new difficulty that full and efficient use of these recommendations requires a deep understanding of their principles. This is even more true of the suggestions of Seshagiri (1972), which achieve efficiency at a heavy cost in simplicity.

The characteristics of the previously published algebraic methods (Fromm, 1970; Orsi, 1972; Indge & Childs, 1976) are the reverse of those of the schematic method: it is much more certain that the result is complete [provided that no products are improperly deleted; see Cornish-Bowden (1976*b*)], but it is more difficult to recognize invalid products. In the method of Fromm (1970), some products are generated more than once, and the repetitions must be recognized and deleted. The method of Orsi (1972) avoids this problem, but increases the number of cycles to be eliminated by failing to ensure that every product contains a route to the target species. (In the terminology of the present paper it does not include a check for rule 5.) Moreover, both authors dismiss too readily the problem of recognizing cyclic products in complex mechanisms. (This problem is, incidentally,

aggravated by the widespread practice of ignoring the recommendations of the Enzyme Commission on the numbering of rate constants: a product that contains, for example, $k_{+3}a \cdot k_{-3}$ is obviously cyclic, but the cyclic character of the same combination may pass unnoticed if it is written as $k_5a \cdot k_6$.) Clearly an acceptable algebraic method should include a reliable rule for the detection of cycles. The method described in the present paper includes such a rule, albeit one that is more laborious to apply than the simple inspection by the schematic method.

Several previous FORTRAN programs for deriving rate equations exist [e.g. Rhoads & Pring, 1968 ('K program'); Hurst, 1967, 1969; Fisher & Schulz, 1969; Rudolph & Fromm, 1971], as well as some others written in other languages or for particular machines [e.g. Rhoads & Pring, 1968 ('D program'); Kinderlerer & Ainsworth, 1976; Indge & Childs, 1976]. In addition, Lam & Priest (1972) have described a program for generating King–Altman patterns, but without providing a complete rate equation. This last is based on the same 'Wang algebra' as the method of Indge & Childs (1976) and presumably leads to the same difficulties if any rate constants appear more than once in the mechanism (Cornish-Bowden, 1976*b*). Most of the other programs are based on the method of King & Altman (1956), though two [Rhoads & Pring, 1968 ('D program'); Hurst (1969)] use the full determinant method described by Kistiakowsky & Shaw (1953) and do not take advantage of the simplifications noted by King & Altman (1956). The determinant method is slightly more general (see Hurst, 1967), but it is also less efficient and leads to much longer execution times. For example, Hurst (1969) reports an execution time of 24.5s on an IBM 360 computer for the mechanism shown in Scheme 1; I obtained 0.4s on an IBM 370 computer. The execution times of several minutes given by Fisher & Schulz (1969) also seem surprisingly long, but this may reflect the use of an earlier generation of computer rather than any inefficiency in their method.

Most of the programs have been described only in outline or in technical terms that would not readily permit the user to mimic them by hand. Consequently it is difficult for the user to modify them to suit special needs, to check their operation manually, or to rewrite them to take advantage of the special features of particular computers. Accordingly, the emphasis in the present paper has been primarily on the details of the procedure and only incidentally on its implementation as a computer program. The program does in fact follow a protocol essentially identical with that described.

I am grateful to Dr. A. C. Storer and Dr. K. F. Tipton for persuading me of the practical usefulness of this work and for suggesting improvements to the first draft. The work was initiated in the Department of Biochemistry, University of California at Berkeley, and some of the final testing of the method, carried out in response to helpful comments of the Referees, was done at the University of Guelph, Ontario, Canada.

References

Ainsworth, S. (1974) *J. Theor. Biol.* **44**, 161–165
Botts, J. & Morales, M. (1953) *Trans. Faraday Soc.* **49**, 696–707
Cha, S. (1968) *J. Biol. Chem.* **243**, 820–825
Childs, R. E. & Bardsley, W. G. (1975) *Biochem. J.* **145**, 93–103
Cornish-Bowden, A. (1976*a*) *Principles of Enzyme Kinetics*, pp. 34–38, Butterworths, London
Cornish-Bowden, A. (1976*b*) *Biochem. J.* **159**, 167
Crabbe, M. J. C. & Bardsley, W. G. (1976) *Biochem. J.* **157**, 333–337
Fisher, D. D. & Schulz, A. R. (1969) *Math. Biosci.* **4**, 189–200
Fromm, H. J. (1970) *Biochem. Biophys. Res. Commun.* **40**, 692–697
Hurst, R. O. (1967) *Can. J. Biochem.* **45**, 2015–2039
Hurst, R. O. (1969) *Can. J. Biochem.* **47**, 941–944
Indge, K. J. & Childs, R. E. (1976) *Biochem. J.* **155**, 567–570
Kinderlerer, J. & Ainsworth, S. (1976) *Int. J. Biomed. Comput.* **7**, 1–20
King, E. L. & Altman, C. (1956) *J. Phys. Chem.* **60**, 1375–1378
Kistiakowsky, G. B. & Shaw, W. H. R. (1953) *J. Am. Chem. Soc.* **75**, 866–871
Lam, C. F. & Priest, D. G. (1972) *Biophys. J.* **12**, 248–256
Meunier, J.-C., Buc, J., Navarro, A. & Ricard, J. (1974) *Eur. J. Biochem.* **49**, 209–223
Orsi, B. A. (1972) *Biochim. Biophys. Acta* **258**, 4–8
Rhoads, D. G. & Pring, M. (1968) *J. Theor. Biol.* **20**, 297–313
Rudolph, F. B. & Fromm, H. J. (1971) *Arch. Biochem. Biophys.* **147**, 515–526
Seshagiri, N. (1972) *J. Theor. Biol.* **34**, 469–486
Storer, A. C. & Cornish-Bowden, A. (1977) *Biochem. J.* **165**, 61–69
Volkenstein, M. V. & Goldstein, B. N. (1966) *Biochim. Biophys. Acta* **115**, 471–477