Calculation of Unbiased First-Order Rate Constants. Acceleration by Overrelaxation with a Responsive Overrelaxation Factor

C. GARDNER SWAIN,* MARGUERITE S. SWAIN, and LAWRENCE F. BERG

Department of Chemistry, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Received May 18, 1979

Shortcomings of previous methods for evaluating first-order rate constants from kinetic data are summarized. KORE, a 92-statement Fortran computer program, is listed and recommended. It evaluates all three parameters (a, b, and k) in the expression for predicted data ($a + be^{-kt}$), also standard deviations and correlation coefficient, has less than half as many statements, and is easier to use than previous programs for this purpose. It illustrates a potentially widely applicable technique for accelerating convergence of simple iterative calculations by repeated automatic adjustment of an overrelaxation factor in each iterative cycle to set it close to its current optimum value, determined from parameter changes in the two most recent cycles.

INTRODUCTION

Purpose. KORE (**k**inetic analysis using **o**ver-**re**laxation) is a 92-statement Fortran computer program to calculate first-order rate constants k from kinetic data (y_t vs. time t) by least-squares fitting of predicted data ($a + be^{-kt}$) to observed data (y_t). All observed data are considered to be equally precise, since this assumption is usually valid for such analytical measurements. Errors in t are assumed to be negligible compared to those in y_t, again the usual situation.

It offers several advantages over previous computer programs for fitting an exponential expression. It has fewer statements by half than any previously published, and is more easily understood or modified because it involves simpler logic. It can cope with almost any observed data, and the user does not have to supply an initial estimate of k. Because it does format-free or list-directed reading of input data, it allows easier punching or typing without concern about the number of data per card or line, or their exact location on a card or line. It converges k to eight figures and calculates a correlation coefficient, in addition to standard deviations for data and for k.

Overrelaxation by a factor repeatedly reevaluated by the program is used to accelerate convergence of the iterative calculations in KORE. Since this novel feature is also widely applicable to iterative computer calculations of other kinds, its illustration is a second major objective of this paper. Simple iterative calculations without overrelaxation usually underestimate parameter changes required in each cycle. Making 60% larger changes (by an overrelaxation factor of 0.6) has often been recommended as a means of achieving convergence in fewer cycles. However, much larger factors (occasionally even over 1000) are calculated and work effectively in our technique, which uses recent past parameter changes to reevaluate a currently nearly optimum factor for each cycle.

Uses of First-Order Rate Constants. First-order rate constants k have facilitated studies of reaction mechanisms for more than a century. In 1850 Wilhelmy integrated the first-order rate law and used its integrated logarithmic form to evaluate a first-order (pseudo-unimolecular) k from polarimetric data on hydrolysis of sucrose catalyzed by various acids.[1] Since then, it has been standard procedure to study complicated reactions under first-order conditions (all reactants but one in large excess), varying the concentration of each reactant in turn over a wide range, to infer the equation of the rate-determining step.[2] For example, the fact that k, derived from $(d[F^-]/dt)/[Ph_3CF]$, is independent of $[NaN_3]$ for reactions of triphenylmethyl fluoride in aqueous acetone solutions, although the yield of Ph_3CN_3 is high (90% for $[Ph_3CF]_0 = [NaN_3]_0 = 0.001\ M$), shows formation of a Ph_3C^+ intermediate to be rate determining.[3] If a higher order dependence on concentrations is found, second- or higher order rate constants may often be calculated from k values by division by one or more concentrations that are nearly constant.[4]

Kinetic methods of analysis for calculating concentrations of reactants can be based on rate constants obtained under reproducible, usually first-order, conditions.[5]

The precision of calculated rate constants may become critical if observed changes due to the variable of interest are less than twofold, as in secondary protium/deuterium isotope effects,[6,7] or if low concentrations or high rates hamper precise observations, as in many biochemical systems. A proper least-squares method can yield significantly more precise and reliable constants than a graphical method.

Previous Methods of Calculation. Graphical methods for evaluating k were relied on from 1931[8] to 1968, but are undesirable for three reasons. First, they are subjective; i.e., a different worker could calculate a k value from the same data differing more than should be expected from the precision of the data. Second, their statistical weighting of the data is improper because they are based on the logarithmic rather than the exponential form of the integrated rate law (see following paragraphs). Third, they provide no quantitative measure of the precision of k.

The commonest graphical method uses one-cycle semilogarithmic graph paper with a horizontal arithmetic time scale. The percent reaction for each datum is plotted, after relabeling the log scale as 0, 10%, 20%, . . ., 90% reaction instead of 1., 0.9, 0.8, . . ., 0.1. If needed, a second cycle relabeled 90-99% is glued or taped on below. The percent reaction is $100(y_0 - y_t)/(y_0 - y_\infty)$, where y_0, y_t, and y_∞ are measured data at the start of the reaction, after time t, and after 10 or more half-lives. A vertical line or bar is drawn through each experimental point with extensions of ±2% reaction, or whatever is the estimated analytical error. Such error lines for late points are relatively long (10 times longer at 90% reaction and 100 times longer at 99% reaction than at 0% reaction). The best straight line is drawn by eye and a transparent straightedge, in an attempt to come vertically closer to those points that have smaller error lines, to try to correct for what is otherwise improper implicit weighting. The k is then $(\ln 2)/t_{1/2}$ or $0.693/(t_{50\%} - t_{0\%})$, where $t_{50\%}$ and $t_{0\%}$ are the times where this line crosses the horizontal lines at 50% and 0% reaction.

Objectivity is better ensured by use of a least-squares method of calculation. To permit a simple, fast, linear least-squares program rather than a more complicated nonlinear least-squares approach, one is tempted to fit the logarithmic form of the integrated first-order rate law by differentiating the equation

$$\sum_t (\ln (y_t - y_\infty) - c + kt)^2 = \text{minimum} \qquad (1)$$

to obtain two equations from which the two unknowns c and

k could be obtained without any iteration.[9-11] Alternatively, c could be taken to be in $(y_0 - y_\infty)$, and k calculated as the only unknown.[12] However, k values obtained by either of these routes are even less reliable than ones obtained graphically, because these calculations make no correction at all for improper implicit weighting. They are biased in favor of late points, i.e., give too much statistical weight to late y_t and final y_∞ data. Although y_t usually has a Gaussian error distribution, $\ln (y_t - y_\infty)$ does not. Equation 1 is still unsatisfactory even if statistical weights are explicitly included to allow for decreasing precision of $\ln (y_t - y_\infty)$ as y_t approaches y_∞. Even with such corrections, by weights decreasing appropriately with increasing percent reaction, it still gives excessive weight to the one datum that affects all terms in the sum, namely y_∞, which is usually no more accurate nor precise than other y_t data. Several informative error discussions have been published.[7,8,13,14]

Fitting an exponential expression (eq 2) for predicted data

$$p_t = a + be^{-kt} \qquad (2)$$

p_t to measured data y_t at the measured times (t) is generally preferable to any of the preceding methods based on logarithmic expressions. This has been recognized by many workers since 1931.[13-16] Furthermore, it was recognized by 1954[17] that digital computers excel analog computers for kinetic and engineering calculations when accuracy, speed, and cost are all considered realistically. Accordingly, numerous digital computer programs were written from 1954 to 1972 to evaluate the parameters (a, b, k) from sets of experimental data, using eq 3 (nonlinear least squares), which automatically

$$\sum_t (y_t - p_t)^2 = \text{minimum} \qquad (3)$$

ensures proper implicit weighting. The two best documented programs are those of DeTar.[16] They permit various options for other kinds of statistical weighting. For example, they permit standard deviations of y_t to be read in individually for each point, if errors in y_t are known to be non-Gaussian. However, LSG, the shorter of them, still has a main routine of more than 200 Fortran statements, plus two subroutines. Moreover, both require the user to supply a reasonable initial k or else to designate data points with equal time spacing from which an initial k can be calculated. We felt that there was a need for a simpler program with fewer than 100 statements that would find an initial k by itself and incorporate implicitly the equal weighting of all data and neglect of errors in t that are appropriate in most situations (eq 2 and 3). KORE was developed to fill this need.

Although the calculation of a and b from all the data removes any disproportionately large influence of y_0 or y_∞ points, this should not, on the other hand, be taken to mean that very early or late points are less important than the others. In fact, it is highly desirable to have such extreme points, as well as six or more points between them approximately evenly spaced in terms of percent reaction.

LOGIC OF KORE

Calculation of the Three Parameters. This method of calculation of a, b, and k is designed to take major advantage of special properties of this exponential function; hence it is not an adaptation of a general nonlinear least-squares program. It begins with rough estimates of a, b, and k, for which it uses the last datum y_n, the difference between first and last data $(y_1 - y_n)$, and $(\ln 2)/(t$ of middle[18] datum$)$, respectively. In the first half of each iterative cycle, it calculates a better k from the current a, b, and k by one step of Newton's method for solving a transcendental equation, i.e., from $k - x'(k)/x''(k)$, where x' and x'' are first and second partial derivatives with respect to k of the function x that we wish to minimize, $\sum_t(a + be^{-kt} - y_t)^2$. In the second half of each iterative cycle it calculates a better a and a better b from the current a, b, and k by solution of the two simultaneous linear equations in these two unknowns obtained by partial differentiations of x with respect to a and b, i.e., by ordinary linear least squares. Thus, this overall nonlinear least-squares curve fitting is accomplished by a succession of linear least-squares steps, one such step in each half cycle. Cycles are repeated until the fractional change in k falls below 10^{-8}, this serving as the criterion of convergence.

Overrelaxation. Overrelaxation is generally useful to accelerate any iterative procedure in which the various parameters are improved alternately or successively rather than simultaneously. If they are improved alternately, the incorrect assumption is made, when calculating each new parameter value, that the values of the other parameters are correct and unchanging. Instead, all of the parameters are usually interrelated and interdependent. As a result of the incorrect assumption, calculated parameter changes are generally too small. Such iterative calculations should still converge to all correct final values eventually, but many-fold more iterative cycles are then required unless overrelaxation is used to compensate. Most simple or unsophisticated iterative calculations are of this alternate improvement sort. Although the above procedure determines a and b simultaneously, it alternates improvement of these with improvement of k; hence it still makes the incorrect assumption. Overrelaxation is therefore used to correct for this.

Overrelaxation of k at the middle of each cycle effectively reduces the number of cycles required for convergence (from an average of 190 cycles to an average of 12 cycles for the datasets that we tested). By overrelaxation of k we mean that k is changed not just by x'/x'' as indicated above but instead by this quotient multiplied by $(1 + f)$, where f is an "overrelaxation factor". This factor is 0.5 in cycle 1 but is reevaluated as the larger of 0.5 or fm in each subsequent cycle, where f is its value in the immediately preceding cycle and its multiplier m is 1, 2, or $^1/_2$ depending on whether d_{i-1}/d_i, the ratio of the preoverrelaxation change in k in the preceding cycle $(i - 1)$ to that in the present cycle (i), is above 4.0, between 4.0 and 0.7, or below 0.7. A d_{i-1}/d_i ratio above 4.0 is considered quite satisfactory progress not requiring any change in f, because such ratios should decrease the change in k by 10^8-fold in 12 or fewer cycles. A ratio between 4.0 and 0.7 represents slower progress that is advantageously accelerated by a doubling of the overrelaxation factor. In this problem, a ratio below 0.5 indicates instability, divergence, or oscillation of k values, which is best corrected by halving f whenever the ratio is below 0.7. The 0.7 value is not critical, because a dividing point of 0.5 or 0.8 worked nearly as well with all the datasets that we tried. In other kinds of iterative programs for problems with more parameters, we have found it better to increase or decrease f by a smaller factor than 2.0, e.g., by a factor of 1.6 or 1.3, but ratios of 4.0 and 0.7 still seem satisfactory for the specified dividing points between no change, increase, or decrease in f.

The last overrelaxation factor is the one used in the next to last cycle. Iterations are terminated when the fractional change in k without overrelaxation becomes smaller than 10^{-8}. This final k is not overrelaxed, because overrelaxation could improve only subsequent a, b, or k values. Both the maximum f used and the last f vary widely and unpredictably from 0.5 to 1024 with different sets of kinetic data. Nevertheless, the increase in rate of convergence resulting from this use of overrelaxation factors calculated from recent changes in k is often spectacular.

Overrelaxation factors could of course be determined automatically by a program in other ways. In fact, we earlier calculated the sum of squared residual differences between observed and calculated data with many different trial values of f to select the optimum f for each cycle. That required fewer cycles for convergence, but unfortunately involved more statements and ran slower because each cycle took much longer to execute. We therefore believe that our present use of only a single f per cycle, determined by recent past changes prior to overrelaxation of a parameter that is subsequently overrelaxed, is significantly more efficient for managing or controlling overrelaxation.

In other problems with more parameters, we have found it helpful to divide parameters into two groups and in each regular cycle to overrelax all the parameters in only one group, always the same group, by a factor determined for that cycle by this method applied to a typical one of the parameters in that group. However, one should replace this variable overrelaxation by more modest overrelaxation by a fixed factor (e.g., 1.6) in the 3–6 cycles immediately preceding any periodic major extrapolation of values of parameters in the other group (no such extrapolation is involved in KORE) and suppress overrelaxation completely in the special cycle following any such extrapolation and in the last cycle which yields the final values. Limiting overrelaxation factors to values not exceeding an empirically determined ceiling may give faster convergence in some problems.

Standard Deviations. Our goal was a final program section of under 20 statements to estimate average random errors in the data and in k. These estimates should each be based on all the data and have a reliability adequate to show the numbers of significant figures that should be retained in the predicted data and in k.

As a reasonable (and usually the only available) estimate of the uncertainty of any y_t or p_t (including p_0 and p_∞), we use the standard deviation of y_t from p_t:

$$s_y \simeq s_p \simeq [\sum_t (y_t - p_t)^2/(2(n-3))]^{0.5} \qquad (4)$$

The factor of 2 assumes equal errors for y_t and p_t, and the degrees of freedom $(n - 3)$ reflect the three parameters in the expression for p_t (eq 2).

To obtain s_k, the standard deviation of k, we use an indirect and novel approach. Since it is difficult to follow the propagation of error from y_t values to k in the absence of an unbiased analytical expression for k, we reverse the process and first use eq 2 to derive the ratio of s_p^2 (mean variance of p_t) to s_k^2 (variance of k), then use the reciprocal ratio to get s_k^2 from s_p^2. We neglect effects of a and b because they are not independent sources of error.

$$\begin{aligned} s_p^2 &\simeq s_k^2 \sum_t (\partial p_t/\partial k)^2/n \\ s_k^2 &\simeq s_p^2 n/\sum_t (\partial p_t/\partial k)^2 \qquad (5) \\ s_k &\simeq s_p\{n/[b^2 \sum_t (te^{-kt})^2]\}^{0.5} \end{aligned}$$

This s_k is only approximate. A much better value is obtainable, if wanted, by modifications described in the following paper.[19] As a percent of k, s_k is usually three to five times s_p as a percent of b. Most s_k values for rate constants in the literature are between 5 and 15% of k. Furthermore, inaccurate data due to systematic (nonrandom) errors sometimes make k values several-fold less reliable than s_k might suggest.[20] However, such errors are generally unknown, any detected having been eliminated or reduced below the random error level. Statistical analysis cannot ferret out such systematic errors.

Correlation Coefficient. This quantity r is the square root of the determination coefficient r^2:

$$r^2 = 1 - ((n-1)\sum_t (y_t - p_t)^2)/((n-3)\sum_t (y_t - \bar{y}_t)^2) \qquad (6)$$

```
      IMPLICIT REAL*8(A-H,K,P-Z);                             1: FORMAT(5X,'
     &KORE,  FITS P=A+B*EXP(-K*T), S.,S.,&BERG, J.C.INF.COMP.SC.,1979'/)
      DIMENSION T(100),T(50),Y(50),E(50)
      DATA NA,NM,HD,CN/1,30,1.D2,1.D-8/;                        WRITE(6,1)
C---READ M & N VALUES FOR ONE SET,  THEN M TITLE LINES & N DATA---------
   3  READ(5,*,END=70)M,N; IF(NA.GT.1)WRITE(6,4)NA;   4: FORMAT(///I37/)
      IF(M.EQ.0) GO TO 6;  L=M*20; READ(5,5)(T(J),J=1,L); 5:FORMAT(20A4)
      WRITE(6,8)(T(J),J=1,L);                6: READ(5,*)(T(J),Y(J),J=1,N)
C---CALCULATE FIRST APPROXIMATIONS FOR K, A & B-------------------------
      K=.693/T((N+1)/2);  A=Y(N);  B=Y(1)-A;  Z=0.;  8: FORMAT(1H ,20A4)
      M=0;          NC=0;  D=0.;  DO 9 J=1,N;  9: Z=Z+Y(J);       GO TO 28
C---CALCULATE BETTER A AND B (FROM OVERRELAXED K)-----------------------
  10  F=0.;     G=0.; H=0.;     DO 16 J=1,N;         F=F+E(J);    G=G+E(J)**2
  16  H=H+E(J)*Y(J); C=N*G-F*F;                  A=(Z*G-F*H)/C;  B=(N*H-F*Z)/C
C---CALCULATE BETTER VALUE FOR K (THE FIRST-ORDER RATE CONSTANT)--------
  20  NC=NC+1; F=0.; G=0.;      DO 24 J=1,N; F=F+T(J)*E(J)*(E(J)*B+A-Y(J))
  24  G=G+T(J)**2*E(J)*(E(J)*B*2.D0+A-Y(J));                           DH=D
      IF(G.EQ.0.D0.OR.NC.EQ.NM)GO TO 60;                               D=F/G
      K=K+D;  CK=DABS(D/K);  IF(CK.LE.CN)GO TO 28
      IF(DH/D.GT.4.D0)GO TO 26;             IF(DH/D.LT.0.7D0)M=MAX0(M-1,0)
      IF(DH/D.GE.0.7D0)  M=M+1;          26: XF=.5D0*2**M;      27: K=K+XF*D
  28  DO 29 J=1,N;    E(J)=0.D0; 29:IF(K*T(J).LT.3.5D1)E(J)=DEXP(-K*T(J))
      IF(NC.EQ.0)GO TO 20;   IF(CK.GT.CN)GO TO 10
C---PRINT DATA (T,Y), PREDICTED DATA (P) & DEVS. (Y-P) IN A TABLE-------
      WRITE(6,31);  W=A+B;            G=0.;  H=0.;   31: FORMAT('0  T = TIM
     &E      OBS. Y OR % RCTN    CALC. P OR % RCTN   DEV. Y-P OR Y%-P%'/)
      DO 33 J=1,N;  PT=A+B*E(J);    X=Y(J)-PT;      G=G+Y(J)**2;    H=H+X*X
      YR=HD*(W-Y(J))/B;  32: FORMAT(F11.4,2(F11.4,F10.3),1PD11.2,0PF9.3)
      PR=HD*(W-PT)/B;  DR=YR-PR;   33: WRITE(6,32)T(J),Y(J),YR,PT,PR,X,DR
C---PRINT K, A, B, A+B, STANDARD DEVS., AND CORRELATION COEF.-----------
      WRITE(6,35)K,A,B,CK,NC,W,XF,NC;    35: FORMAT(//'OK =',1PD11.4,3X,'A
     & =',D11.4,3X,'B =',D11.4,8X,'P= A+B*EXP(-K*T)'/' CK=',D8.1,' = FRA
     &CTIONAL CHANGE IN K IN CYCLE',I3,8X,'P(O)=',D11.4/  ' XF=',0PF6.1,
     &1X,'  =  OVERRELAXATION FACTOR IN CYCLE',I3,8X,'P(INFINITE T)= A')
      IF(N.LE.3)GO TO 60; S=DSQRT(H/(N+N-6)); F=0;EV=S*S*2*(N-1)/(G-Z/N)
      R=DSQRT(1.-EV); X=HD*S/DABS(B); DO 36 J=1,N; 36:F=F+(T(J)*E(J))**2
      SK=DSQRT(N/F)*S/DABS(B); V=HD*SK/K;         WRITE(6,37)SK,V,R,S,X,N,EV
  37  FORMAT(' STAND. DEV. FOR K =',1PD8.1,' =',0PF7.3,' % OF K',5X,'R=C
     &ORREL.COEF.=',F9.7/' ST.DV. FOR Y OR P =',1PD8.1,' =',0PF7.3,' % O
     &F B',5X,'N=',I2,4X,'1.-R*R =',1PD8.1)
  39  FORMAT('OK IS MISCALCULATED.       DATA TOO FEW OR MALDISTRIBUTED')
C---RETURN TO STATEMENT LABELED 3 TO LOOK FOR NEXT DATASET--------------
  60  IF(G.EQ.0.D0.OR.NC.EQ.NM)WRITE(6,39); NA=NA+1; GOTO3; 70:STOP; END

C$ENTRY (or appropriate input data marker for compiler used)
4 10
TEST DATASET.   KINETICS OF DEDIAZONIATION OF 9.02E-4 M. PARA-DEUTERATED
BENZENEDIAZONIUM FLUOBORATE IN AQ. 1.00 M. SULFURIC ACID AT 25.00DEG.C.,
T=SEC., Y=ABSORBANCE AT 295 NM., DATA FROM RUN NO. 101 BY JOHN E. SHEATS
TABLE 73, P. 192 OF THESIS CITED IN J.AM.CHEM.SOC.,97(1975),P. 789,REF.2
.00 1.695 .60 1.654 2.64 1.524 6.18 1.32 13.92 .966 19.38 .774
26.94 .570 37.68 .368 84.12 .056 107.34 .021
```

Figure 1. Watfiv version of KORE with test dataset[22] (set no. 1).

```
run 101

 T = TIME      OBS. Y OR % RCTN    CALC. P OR % RCTN   DEV. Y-P OR Y%-P%

    0.0000      1.6950     0.006     1.6951     0.000  -9.80d-05    0.006
    0.6000      1.6540     2.423     1.6545     2.396  -4.64d-04    0.027
    2.6400      1.5240    10.087     1.5235    10.120   5.48d-04   -0.032
    6.1800      1.3200    22.114     1.3202    22.101  -2.33d-04    0.014
   13.9200      0.9660    42.985     0.9653    43.025   6.72d-04   -0.040
   19.3800      0.7740    54.305     0.7740    54.306   2.62d-05   -0.002
   26.9400      0.5700    66.332     0.5699    66.336   6.66d-05   -0.004
   37.6800      0.3680    78.241     0.3689    78.189  -8.75d-04    0.052
   84.1200      0.0560    96.635     0.0556    96.661   4.42d-04   -0.026
  107.3400      0.0210    98.699     0.0211    98.694  -8.37d-05    0.005

K = 4.0413d-02   A =-1.0742d-03   B = 1.6962d+00          P= A+B*EXP(-K*T)
CK= 2.8d-09 = FRACTIONAL CHANGE IN K IN CYCLE 10          P(O)= 1.6951d+00
XF=   2.0   =  OVERRELAXATION FACTOR IN CYCLE 10          P(INFINITE T)= A
STAND. DEV. FOR K = 3.9d-05 =  0.096 % OF K     R=CORREL.COEF.=0.9999999
ST.DV. FOR Y OR P = 3.8d-04 =  0.022 % OF B     N=10     1.-R*R = 2.4d-07
```

Figure 2. Output from test dataset.

where $\bar{y}_t$ is the mean of all y_t.

Correlation coefficients for kinetic measurements are being reported with increasing frequency,[21] because they are the most meaningful measures of random (nonsystematic) errors. They are preferable to standard deviations as a measure of overall goodness of fit because they take into account the range of the data and the number of data. A correlation coefficient r below 0.98 $(1 - r^2 > 0.04)$ indicates either that eq 2 is of the wrong form or that the data are intolerably imprecise by ordinary chemical standards. The high correlation coefficient (0.999 999 9) of our test dataset (Figure 1) suggests that these actual experimental data[22] do correspond to first-order kinetics and are of high quality.

OUTPUT OF KORE

Figure 2 shows the output generated from the input[22] of Figure 1. For each kinetic run analyzed in a given job, the printed output includes its chronological order number in the job (by an integer, centered) for each run beyond the first, its title (user-supplied, 0 to 5 lines), and a table showing, in a separate row for each of its points, the time (T = t, user supplied), the observed datum expressed both absolutely (Y = y_t, user supplied) and as the Y% reaction $(100(y_t - a - b)/(-b))$, the predicted datum expressed both absolutely (P = p_t, eq 2) and as the P% reaction $(100(p_t - a - b)/(-b))$, and

Table I. Parameters (Rounded) Calculated from Synthetic Input Data by KORE

	set no.					
	2		3		4	
	t	y_t	t	y_t	t	y_t
Input data	0.	10.	0.	10.	0.	10.
	1.	3.	1.	2.	0.	2.
	2.	4.	2.	5.	2.	1.
	5.	1.	3.	1.	2.	6.
			1000.	0.	4.	1.
					4.	5.
Output						
k	1.58		1.35		0.80	
a	2.03		1.32		2.88	
b	7.90		8.49		3.13	
r	0.947		0.883		0.680	
NC	13		14		13	
XF	2.		16.		2.	

the deviation or difference between observed and predicted data expressed both absolutely (Y–P) and as the difference between the Y and P percentages ($100(y_t - p_t)/(-b)$). It then records K = k, A = P(INF) = $p_\infty = a$, B = b, and P(0) = $a + b$, each to five figures, also the fractional change in K in the last iterative cycle (CK $\leq 10^{-8}$), the number of the last cycle (NC $\leq$ 30), and the value of the last overrelaxation factor (XF = f).

For any analysis based on more than three points it also lists the standard deviation of k (SK = s_k, eq 5) both absolutely and as a percentage of k, the standard deviation for data ($s_y = s_p$, eq 4) both absolutely and as a percentage of $|b|$, the overall correlation coefficient (R = r, ≤ 1, from eq 6), the fraction of the observed variations not accounted for ($1 - r^2$, the fraction due to random errors or perturbing factors not explicitly allowed for by eq 2), and the number of points (N = n).

Table I summarizes input and output for three other datasets that we have found useful as critical tests of KORE and other programs for calculating k values. Whereas set 1 was actual good experimental data,[22] 2–4 are shorter synthetic sets with enormous scatter, hence are a greater challenge for any program. They probably provide a more stringent test than necessary because any kinetic data this bad should surely be replaced by better data.

KORE yields converged parameters from sets 2–4, each in 14 cycles or less. LSG[16] does not converge with sets 2–4 even after adding statements to avoid underflow, increasing the allowed number of cycles to 25, and using the same initial parameters as those automatically calculated by KORE; sets 2 and 4 diverged, probably owing to poor initial k or large scatter, and set 3 was still far from convergence after 25 cycles. On the other hand, DeTar's programs[16] have advantages for handling other datasets where errors in y_t are unequal or non-Gaussian or where errors in t are significant.[16]

KORE handles very late data correctly. The late time (1000.) in set 3 causes no underflow or other computational complication. Another synthetic set consisting of the four data, (y at t) 9.9 at 2., 9.99 at 3., 9.999 at 4., and 9.9999 at 5., converges correctly to k = 2.3026 (ln 2), a = 10., b = −10. with r = 1.000 000 0 at NC = 21 cycles with a maximum f of 1024 and last f (XF) of 512. Obviously, KORE must fail if data are unsuitable, e.g., linear in time, or include only two different y_t values or only two different t values. The message "k wrong because data maldistributed" is then printed, and execution is switched to the next dataset, or stopped if no more data follow.

INSTRUCTIONS FOR USE OF KORE

Coding and Compilation. Figure 1 is a Fortran IV program for a Watfiv compiler,[23] which permits multiple statements on a line, so that we can include all 92 statements and 7 comments on 45 lines, listable in minimum journal space. For the IBM G1 Fortran compiler, retype with only one statement per line and no colons or semicolons. For the Honeywell Multics Fortran compiler, make all labeled statements begin a line (with no colon), replace asterisks in statements labeled 3 and 6 by 2, and add a statement, 2 format(v), to specify the list-directed reading. Other Fortran compilers may require a few other changes. "Hunt and peck" typing time for whichever compiler is preferred is less than 90 min.

If any run to be processed contains more than 50 points, the dimensions of T, Y, and E must be increased from 50 to a number at least as large in the third (dimension) statement.

Double precision (real*8) is specified in the first statement because single precision leads to undamped oscillating k values and less than five significant figures when either LSG or KORE is run on an IBM 360 or 370 computer. With any computer having real single precision of $\geq$8 decimal digits (e.g., Honeywell 6180 Multics), one could delete *8, specify 1.E-7 for CN in the data statement, and change all constants and functions to single precision (e.g., 0.D0 to 0., DEXP to EXP). However, there is nothing to be gained by single precision unless the compiler used cannot supply double precision, because execution time with double precision ($\geq$16 decimal digits) is so short (see last paragraph of next section).

Data Input and Execution. Many unrelated datasets (kinetic runs) may be processed in a single job, and are simply stacked one after another in the data section (following the program). The data for each of them must comprise one control card, then from zero to five cards bearing a suitable title, legend, comments, and/or identification numbers, then the cards with t and y data. No blank cards or separator cards are used. (If data are entered on a time-sharing terminal, replace "cards" by "lines".)

Each control card is punched with two integers (or a line is typed with two integers) in any desired columns separated by one or more blanks: the first integer specifies the number (0–5) of title cards or lines immediately following (for that run only); the second (0–50) specifies the number of points (pairs of t, y_t values) immediately following the 0–5 title lines. Data on subsequent lines must be real numbers (with decimal points) with each t followed by its y_t, separated from each other and from other pairs by one or more blanks (or a comma) for format-free or list-directed reading. As many t and y_t values as will fit may be punched or typed on each line, but no value should be split between two lines.

Computer time for batch processing the four runs of Figure 1 and Table I with a Watfiv V1L4 compiler on an IBM 370-168 computer was 0.19 s (0.05 for compilation, 0.14 for execution). Computer execution time for these same data processed via a time-sharing typewriter terminal with compilation by a Honeywell 6180 Multics new (1977) standard Fortran compiler was 0.66 s. Evidently execution time is no longer important with this program because it is so fast.

ACKNOWLEDGMENT

This work was supported in part by a research grant from the Donors of the Petroleum Research Fund administered by the American Chemical Society. Mr. Berg was aided by computer charge credits by the MIT Student Information Processing Board.

REFERENCES AND NOTES

(1) L. Wilhelmy, *Ann. Phys. Chem.* (*Leipzig*), **81**, 413, 499 (1850).
(2) A. V. Harcourt and W. Esson, *Phil. Trans. Roy. Soc.* (*London*), **157**, 117 (1867).
(3) C. G. Swain, C. B. Scott, and K. H. Lohmann, *J. Am. Chem. Soc.*, **75**, 136 (1953).

(4) E.g., J. Rocek and T.-Y. Peng, *J. Am. Chem. Soc.*, **99**, 7624 (1977); E. A. Moelwyn-Hughes, "The Chemical Statics and Kinetics of Solutions", Academic Press, New York, 1971, pp 124-133.
(5) H. B. Mark, Jr., and G. A. Rechnitz, "Kinetics in Analytical Chemistry", Interscience, New York, 1968, 339 pp.
(6) K. Humski, V. Sendijarević, and V. J. Shiner, Jr., *J. Am. Chem. Soc.*, **95**, 7722 (1973).
(7) C. J. Collins, *Adv. Phys. Org. Chem.*, **2**, 62-74 (1964).
(8) W. E. Roseveare, *J. Am. Chem. Soc.*, **53**, 1651-1661 (1931).
(9) J. Casanova, Jr., and E. R. Weaver, *J. Chem. Educ.*, **42**, 137-139 (1965).
(10) R. C. Williams and J. W. Taylor, *J. Chem. Educ.*, **47**, 129-133 (1970).
(11) K. B. Wiberg in *Tech. Chem. (N.Y.)*, **6**, 746 (1974).
(12) T. R. Dickson, "The Computer and Chemistry", W. H. Freeman, New York, 1968, pp 160-161.
(13) L. J. Reed and E. J. Theriault, *J. Phys. Chem.*, **35**, 673-689, 950-971 (1931); C. L. Perrin, "Mathematics for Chemists", Wiley, New York, 1970, pp 162-3.
(14) D. F. DeTar, *J. Chem. Educ.*, **44**, 759-761 (1967); S. Wold, *Acta Chem. Scand.*, **21**, 1986 (1967); D. F. DeTar, ref 16 pp 139-142; *ibid*, Vol. 4, 1972, pp 88-95.
(15) P. Moore, *J. Chem. Soc., Faraday Trans. 1*, **68**, 1890 (1972), another exponential program, but written in ALGOL, with no published listing and an unspecified number of statements.
(16) D. F. DeTar, "Computer Programs for Chemistry", Vol. 1, W. A. Benjamin, New York, 1968, FORTRAN programs LSG, pp 117-125, and LSKIN1, pp 126-173.
(17) J. A. Beutler, *Chem. Eng. Prog.*, **50**, 569 (1954).
(18) KORE picks $(n + 1)/2$ as the middle datum (integer division, neglecting any remainder).
(19) C. G. Swain, M. S. Swain, and P. F. Strong, following paper in this issue.
(20) An extreme example is a precisely determined published rate constant with an error of more than five powers of ten caused by a systematic error, cited by C. G. Swain and R. E. Davis, *J. Am. Chem. Soc.*, **82**, 5949 (1960), ref 5
(21) I. Szele and H. Zollinger, *J. Am. Chem. Soc.*, **100**, 2811 (1978).
(22) These kinetic data were obtained by ultraviolet spectrometry. *Cf.* C. G. Swain, J. E. Sheats, K. G. Harbison, D. G. Gorenstein, and R. J. Rogers, *J. Am. Chem. Soc.*, **97**, 783-800 (1975).
(23) P. Cress, P. Dirksen, and J. W. Graham, "Fortran IV with WATFOR and WATFIV," Prentice-Hall, Englewood Cliffs, N.J., 1970.

A Simple, Reliable, and Fast Monte Carlo Method for Estimating the Uncertainty in Any Computer-Calculated Quantity

C. GARDNER SWAIN* and MARGUERITE S. SWAIN

Department of Chemistry, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

PETER F. STRONG

Arthur D. Little, Inc., Cambridge, Massachusetts 02140

Received May 18, 1979

UNCERT is a programmable method for evaluating the uncertainty (standard deviation) of any computer-calculated quantity, which is often easier to apply and freer of questionable assumptions than conventional propagation-of-error analyses. It utilizes a Monte Carlo strategy of multiple random trials, i.e., recalculations from slightly and randomly altered input data. Fifty trials generally yield a standard deviation with a relative precision (standard deviation) better than ±10%, which is more than adequate for most purposes. The method is illustrated by its application as an addendum to the first-order rate constant program of the preceding paper (with L. F. Berg).

INTRODUCTION

This programmable analytical method, which we call "UNCERT", was designed to determine uncertainties (standard deviations) of any quantities that are calculated from input data subject to random experimental errors or noise. It clearly excels standard propagation-of-error analyses when calculated quantities are determined iteratively (by successive approximations) owing to lack of explicit analytical expressions, or when standard error analyses allowing for interdependence (covariance) of the calculated quantities become too complicated and time-consuming. UNCERT is therefore potentially useful in computer calculations in many fields of science, engineering, and management. Scarcely any quantity is worth calculating without also establishing its uncertainty with a relative precision of ±50% or better. UNCERT quickly gives uncertainties with ±10% precision.

UNCERT is simpler in program logic and required coding than other methods of error analysis. It involves no matrix inversion nor even any calculation of a partial derivative. Instead it employs Monte Carlo recalculations of all the final calculated quantities many times from slightly and randomly altered input data. Then a separate standard deviation for each calculated quantity is derived from its root-mean-square variation in these random trials. UNCERT avoids biases that occur in other methods of estimating errors because they neglect higher order interactions and sometimes even first-order interactions among the calculated quantities. UNCERT is thus an acronym for "**un**biased **c**alculation of **er**rors by random **t**rials".

Monte Carlo solutions have been used in many other kinds of problems,[1-5] but only very rarely for error analysis.[6] Most treatments of error analysis overlook this approach entirely.[7] However, the following unique advantages make the Monte Carlo method even more desirable in error analysis than in other applications. First, the number of trials does not increase with the number of unknowns, and coding and execution time per trial increase only as the first power of the number of unknowns (quantities or parameters calculated), whereas the number of derivatives that had to be calculated by previous methods increased with the square of the number of unknowns. Therefore, the Monte Carlo method is simpler and faster than other methods in error analyses involving many unknowns. Second, the coding used to obtain the uncertainties is essentially just the coding used to solve for these unknowns initially. This is especially important in nonlinear problems, where the special extra coding required by other methods of error analysis is often long and complicated. It is also true, for reasons explained below, that each trial of the Monte Carlo method is typically much faster than the initial calculation, and is sometimes faster by as much as a factor of 50. Therefore,