Comput. Biol. Med. Pergamon Press 1977. Vol. 7, pp. 231–247. Printed in Great Britain.

CONSTANT-STEP APPROXIMATION OF MULTI-EXPONENTIAL SIGNALS USING A LEAST-SQUARES CRITERION

R. VAN MASTRIGT
Department of Urology, Erasmus University Rotterdam, P.O. Box 1738, Rotterdam, The Netherlands

(*Received* 22 *April* 1974; *in revised form* 15 *April* 1976)

Abstract—A FORTRAN IV computer program is presented which fits up to three exponential terms and a constant to experimental data according to a least-squares criterion.
Initial estimates are necessary, but no specific kind of spacing is required.
The program was tried out on artificially generated three-exponential curves with added white noise. The parameters required could be determined with satisfactory accuracy. The program uses a considerable amount of CPU time, but can be run on a mini-computer.

Exponential Fitting Sum of least squares Stepwise

INTRODUCTION

Many physical and biophysical phenomena (e.g. radio-active decay, responses of metabolic systems to external stimuli and stress relaxation in visco-elastic materials [1] yield a signal, consisting of the sum of a number of exponential terms plus a constant. Analysis of these phenomena often requires separation of the signals into their components. Basically, three methods can be distinguished:

(a) The peel-off method. The signal is plotted semi-logarithmically, and the parameters of the slowest exponential are estimated from the tail of the signal, where other components will be almost zero. This slowest component is then subtracted from the signal and the method is repeated [2].

(b) The analog method. The signal is projected on a screen, together with a multi-exponential signal derived from an R–C network. The parameters of this network are changed manually, until the curves match [3].

(c) The digital method. The signal is sampled, digitized and fed to a digital computer, which fits a sum of exponentials to it, by a least-squares approximation method.

Provided a good computer program is available, the last method is the most convenient and often the most accurate; we therefore developed a method to analyse a set of experimental visco-elastic decay-curves. Since the computer program used for this purpose appears to be of more general interest, we are describing it separately in this paper.

STATEMENT OF THE PROBLEM

Given a number of experimental points (x_i, y_i) the problem is to find the parameters (a_j, γ_j) such that

$$\sum_{i=1}^{N}\left(\sum_{j=1}^{k} a_j \exp[-(\gamma_j x_i)] + a_0 - y_i\right)^2 = \text{minimum}.$$

Because of the exponential terms, the equations to be solved in deriving the parameters (Gauss equations) are non-linear. Now it is generally acknowledged that the solution of these equations is an awkward problem and that in practice it might even lead to a non-unique set of parameters (a_j, γ_j) [4].

It is clear, on the other hand, that the "awkwardness" of the problem must be dependent on the properties of the set of experimental points (x_i, y_i), since e.g. the separation of two exponentials with relaxation constants $\gamma_1 = 1$; $\gamma_2 = 1000$ should be a very simple matter. In this example, the two exponential terms can almost be measured independently, one in the range from $x = 0$ to $x = 0.01$ (where the "faster" exponential is practically zero) and one in the range from $x = 0.01$ (where the slower exponential has in practice not yet shown any change) to $x = 10$. Thus the decision as to whether separation into the desired number of exponential terms is possible must be taken afresh, on the merits of the individual case, for each set of experimental points (x_i, y_i) (or any similar set of data pairs).

The specific problem we want to discuss here concerns visco-elastic relaxation curves measured on urinary bladders [1] or bladder-wall strips [5].

CONSTANT STEP APPROXIMATION

Since the desired computer program had to be run on a mini-computer, it had to be relatively short.

Now we have already mentioned in the previous Section that the equations to be solved in order to determine the parameters are non-linear. They are however *only* non-linear in the exponents γ_j. This means that if these exponents are fixed, the optimum coefficients a_j can be obtained by simple matrix inversion, using the classical least squares method. We can therefore use an iterative method for the exponents only. The simplest method would be to perform iterative steps in the (exponent) parameter space, calculate the optimum coefficients and constant for each point, determine the sum of least squares and then choose the point with the lowest sum of least squares.

Apart from its simplicity, this method has the advantage of giving a clear insight in the form of the minimum and thus supplying the user with an impression of the possibility of separation into exponentials. Of course it is impossible to investigate the entire parameter space in practice, simply because it contains an infinite number of points. We therefore need an initial estimate of the exponents. However, the use of a certain sampling rate and number of experimental points (x_i, y_i) also implies a foreknowledge of the order of magnitude of the exponents (time constants), so this is not a real limitation.

The search can then be confined to evaluating a series of points in the vicinity of the initial estimate. Depending on the results, the "search area" can then be displaced and the procedure repeated until a minimum is reached. It is clear that there will be an uncertainty in the parameters depending on the distance between the points under investigation. This uncertainty is exactly defined and must be chosen by the user, on the basis of the accuracy required and the time available.

OPERATIONAL INTERPRETATION

Introduction

The procedure can be very simple: starting with the initial estimates of the exponents γ_j, take a step in a number of directions, calculate for each step the corresponding coefficients, constant and sum of least squares and choose the best direction. This procedure is then repeated. One problem is how many and which directions should be tried. The minimum number is $2k$, where k is the number of exponential terms. In this case one exponent is varied one step forward and one step backwards, while the others are fixed. The search in the parameter space thus takes place in the axial directions only; it turned out that with our data the minimum could always be found in this way. In other applications it may be necessary to increase the number of directions by varying more exponents at a time.

To speed up the search procedure, the search is started with relatively large steps, which are halved every time the sum of least-squares increases, until the defined minimum step is reached. The rules for this procedure are comprised in what we call "algorithm A". Though it gives a clear insight into the shape of the minimum, it is not very economical because it involves some unnecessary steps.

One of these is the step in the direction we just came from, furthermore, if a lower sum of least squares can be obtained by increasing a particular exponent, there is no need to investigate the *decrease step* too for if this step were also to yield a lower sum of least squares, (which implies that we are on a hill, with a minimum on both sides) there is no way to tell which minimum is "better" or "the real one". We might as well pick one at random by always omitting the other step if a decrease is found in one direction. In practice the "hill situation" has never occurred during work with algorithm A on our data. Finally algorithm A also investigates the central point from which we make the steps. Since this point was chosen as the best step in the previous iteration, the results for it are already known, so they don't have to be determined again.

These three refinements, which almost halved the CPU time are realized in algorithm B. Finally another approach which could be useful, especially when the initial estimates are far from the minimum, is laid down in algorithm C. This performs the same steps as algorithm B, except that when the best direction has been decided on after one complete iteration, it goes on with the same step size in this direction until the sum of least squares starts to rise again. Then all directions are investigated again, and so on. Comparing the three algorithm's from a theoretical point of view, we may state that A gives the clearest insight into the shape of the minimum, and is also the simplest, B works more or less like A but is about twice as fast and gives a less clear impression of the form of the minimum; and C is faster than B when the initial estimates are very bad, but slower than B (because at least one extra step is tried in each iteration) when the initial estimates are good. In our case it turned out that the advantages of algorithm C were hardly ever used; we found algorithm B the most convenient. For testing purposes we used an algorithm as laid down in CACM No. 315 which involves a damped Newton-series iteration.

The FORTRAN program used is divided into a main program and two subroutines, STEP and LIN, which will now be discussed briefly in turn.

The MAIN *program*

The MAIN program (Fig. 1) only provides input and output facilities and since it will have to be rewritten by each user, all READ and WRITE statements are replaced by descriptive COMMENT statements. The input should contain the x and y data, in the arrays $U(I, 6)$ and $U(I, 5)$, and the number N of experimental points in these arrays (maximum 500). There is no need for a special kind of spacing between the points.

Further, the number of exponential terms k (which may be 1, 2 or 3) should be given. The search is started from the initial estimates of the exponents $V(I)$, with steps $G*D(1)$, $G*D(2)$, $G*D(3)$.

Each time the sum of least squares rises in all directions, i.e. when a minimum is reached, G is halved until it equals one. As a safety measure, a maximum number of iterations IRMAX must also be specified.

We used:

$$D(1) = \frac{V(1)}{50}; \quad D(2) = \frac{V(2)}{50}; \quad D(3) = \frac{V(3)}{50}; \quad G = 8; \quad \mathrm{IRMAX} = 50.$$

The constant C(K1), the coefficients C(I), the exponents B(I) and the sum of the least squares PHI can be printed out after the return from the STEP subroutine (statement

```
      COMMON U(500,6),C(6),K,N,PHI,B(4),G,D(3),IRMAX,K1,V(4)              MAIN0100
C     READ THE INITIAL ESTIMATES FOR THE EXPONENTS V(I)                   MAIN0200
C     READ THE MINIMUM STEPS IN THE EXPONENTS D(I)                        MAIN0300
C     READ THE FACTOR G,WHICH DEFINES THE MAXIMUM STEPS IN THE            MAIN0400
C        EXPONENTS G*D(1),G*D(2) AND G*D(3);G SHOULD BE A POWER OF TWO MAIN0500
C     READ THE MAXIMUM ALLOWED NUMBER OF ITERATIONS IRMAX                 MAIN0600
C     READ THE NUMBER OF EXPONENTIAL TERMS K,MAXIMUM THREE                MAIN0700
      K1=K+1                                                              MAIN0750
C     READ X-DATA IN THE ARRAY U(I,6)                                     MAIN0800
C     READ Y-DATA IN THE ARRAY U(I,5)                                     MAIN0900
C     DETERMINE THE NUMBER OF DATA POINTS N                               MAIN1000
      CALL STEP                                                           MAIN3200
C     WRITE THE CONSTANT C(K1)                                            MAIN3300
C     WRITE THE COEFFICIENTS C(I)                                         MAIN3400
C     WRITE THE EXPONENTS B(I)                                            MAIN3500
C     WRITE THE SUM OF LEAST SQUARES  PHI                                 MAIN3600
C     WRITE THE FITTED CURVE U(I,4)                                       MAIN3700
      STOP                                                                MAIN3800
      END                                                                 MAIN3900
```

Fig. 1. Listing of MAIN program.

MAIN 3200). The fitted curve is found in the array U(I,4) and the x and y data are still in U(I,6) and U(I,5).

The STEP *subroutine for algorithm A*

The STEP subroutine (Fig. 2) governs the iteration. The different sets of exponents are formed in the statements STEP0210 up to and including STEP0300. For the sake of simplicity, the central point in the parameter space (which was chosen as the best in the previous iteration and is thus already known) is also included, so $2k + 1$ sets

```
      SUBROUTINE STEP                                                     STEP0000
C
C          VERSION A
C
      COMMON U(500,6),C(6),K,N,PHI,B(4),G,D(3),IRMAX,K1,V(4)              STEP0001
      DIMENSION BB(3,7),PHIT(7)                                           STEP0002
    8 FORMAT (1H ,I3,3F10.5,F15.2,I10)                                    STEP0003
    9 FORMAT(1H0,'NR',11X,'EXPONENTS',20X,'PHI',12X,'JTE')                STEP0004
   10 FORMAT(1H0)                                                         STEP0005
      K2=2*K+1                                                            STEP0006
      AKLE=0                                                              STEP0060
      DO 14 I=1,K                                                         STEP0063
   14 B(I)=V(I)                                                           STEP0066
      DD=G                                                                STEP0068
      WRITE (2,9)                                                         STEP0090
      DO 7 IR=1,IRMAX                                                     STEP0100
      JTE=1                                                               STEP0200
      DO 1 J=1,K2                                                         STEP0210
      DO 1 I=1,K                                                          STEP0220
    1 BB(I,J)=B(I)                                                        STEP0230
      DO 19 J=2,K2                                                        STEP0260
      I=J/2                                                               STEP0270
      INT=1                                                               STEP0280
      IF(2*I.EQ.J)INT=-1                                                  STEP0290
   19 BB(I,J)=BB(I,J)+INT*DD*D(I)                                         STEP0300
      DO 3 J=1,K2                                                         STEP0320
      DO 4 I=1,K                                                          STEP0330
    4 B(I)=BB(I,J)                                                        STEP0340
      CALL LIN                                                            STEP0350
      PHIT(J)=PHI                                                         STEP0360
      IF (PHI.LT.AKLE) JTE=J                                              STEP0370
      AKLE=PHIT(JTE)                                                      STEP0380
      DO 13 I=K1,4                                                        STEP0383
   13 B(I)=0.                                                             STEP0386
    3 WRITE    (2,8) IR,(B(I),I=1,3),PHI,JTE                              STEP0390
      WRITE (2,10)                                                        STEP0400
      DO 6 I=1,K                                                          STEP0430
    6 B(I)=BB(I,JTE)                                                      STEP0440
      IF (JTE.NE.1) GOTO 7                                                STEP0450
   11 IF (DD.EQ.1) GOTO 12                                                STEP0455
      DD=DD/2                                                             STEP0460
    7 CONTINUE                                                            STEP0465
   12 CALL LIN                                                            STEP0470
      RETURN                                                              STEP0480
      END                                                                 STEP0490
```

Fig. 2. Listing of STEP subroutine, algorithm A.

```
      SUBROUTINE STEP                                                   STEP0000
C
C         VERSION B
C
      COMMON U(500,6),C(6),K,N,PHI,B(4),G,D(3),IRMAX,K1,V(4)            STEP0001
      DIMENSION BB(3,7)                                                 STEP0002
    8 FORMAT(1H ,I3,3F10.5,F15.2,I10)                                   STEP0003
    9 FORMAT(1H0,'NR',11X,'EXPONENTS',20X,'PHI',12X,'JTE')              STEP0004
      K2=2*K+1                                                          STEP0006
      DO 14 I=1,K                                                       STEP0063
   14 B(I)=V(I)                                                         STEP0066
      DD=G                                                              STEP0068
      CALL LIN                                                          STEP0070
      AKLE=PHI                                                          STEP0080
      JTE=1                                                             STEP0085
      IR=0                                                              STEP0091
      WRITE (2,9)                                                       STEP0090
      DO 17 I=K1,4                                                      STEP0092
   17 B(I)=0.                                                           STEP0093
      WRITE(2,8)IR,(B(I),I=1,3),PHI,JTE                                 STEP0096
      DO 7 IR=1,IRMAX                                                   STEP0100
      JTEV=JTE                                                          STEP0150
      JTE=1                                                             STEP0200
      DO 1 J=1,K2                                                       STEP0210
      DO 1 I=1,K                                                        STEP0220
    1 BB(I,J)=B(I)                                                      STEP0230
      DO 19 J=2,K2                                                      STEP0260
      I=J/2                                                             STEP0270
      INT=1                                                             STEP0280
      IF(2*I.EQ.J)INT=-1                                                STEP0290
   19 BB(I,J)=BB(I,J)+INT*DD*D(I)                                       STEP0300
      PHIT=AKLE                                                         STEP0315
      DO 3 J=2,K2                                                       STEP0320
      IF ((JTEV/2*4-JTEV+1).EQ.J) GOTO 3                                STEP0325
      DO 4 I=1,K                                                        STEP0330
    4 B(I)=BB(I,J)                                                      STEP0340
      CALL LIN                                                          STEP0350
      IF (PHI.GE.PHIT) GOTO 3                                           STEP0351
      IF (PHI.GE.AKLE) GOTO 16                                          STEP0370
      JTE=J                                                             STEP0371
      AKLE=PHI                                                          STEP0380
   16 IF (J/2*2.EQ.J) J=J+1                                             STEP0381
    3 CONTINUE                                                          STEP0395
      DO 6 I=1,K                                                        STEP0430
    6 B(I)=BB(I,JTE)                                                    STEP0440
      WRITE(2,8)IR,(B(I),I=1,3),AKLE,JTE                                STEP0445
      IF (JTE.NE.1) GOTO 7                                              STEP0450
   11 IF (DD.EQ.1) GOTO 12                                              STEP0455
      DD=DD/2                                                           STEP0460
    7 CONTINUE                                                          STEP0465
   12 CALL LIN                                                          STEP0470
      RETURN                                                            STEP0480
      END                                                               STEP0490
```

Fig. 3. Listing of STEP subroutine, algorithm B.

are generated. The coefficients, constant and PHI for each set are computed by calling subroutine LIN (statement STEP0350).

During each step, one row of output is printed containing the serial number of the iteration, the exponents, PHI and the serial number of the set of exponents with the smallest PHI during this iteration, in that order. If the last-mentioned number (JTE) equals one, the central point was the best, i.e. PHI increases in all directions and if G equals one, iteration is now stopped (STEP0455); otherwise, G is halved (STEP0460) and the process repeated. The coefficients and constant corresponding to the "best" set of exponents must then be computed again (STEP0470), as they were lost during the trying of other sets of exponents.

The STEP *subroutine for algorithm B*

Algorithm B incorporates the following changes compared with algorithm A (Fig. 3); two variables have been added: JTEV (STEP0150) contains the step previously chosen, and PHIT (STEP0315) contains the corresponding sum of least squares. STEP0325 is used to decide whether the step to be taken is a retrograde one (in the direction we came from); if so, it is skipped.

Succesful steps are investigated in STEP0381. If JTE is even which means that the next step will be in the opposite direction, the next step is skipped.

Finally it was possible to omit the evaluation of the central point by changing STEP0320; this made some extra statements (STEP0070–0096) necessary in order to evaluate the value of the central point of the first iteration step (initial estimate). Finally displacement of the printing statement (from STEP0390 to STEP0445) reduced the volume of printout (only one row per iteration instead of one row per iteration step).

The STEP *subroutine for algorithm* C

In this version (Fig. 4), after the normal iteration which equals the scheme in algorithm B, first the value of the winning step is determined (STEP0440) and then again such

```
      SUBROUTINE STEP                                                   STEP0000
C
C          VERSION C
C
      COMMON U(500,6),C(6),K,N,PHI,B(4),G,D(3),IRMAX,K1,V(4)            STEP0001
      DIMENSION BB(3,7)                                                 STEP0002
    8 FORMAT(1H ,I3,3F10.5,F15.2,I10)                                   STEP0003
    9 FORMAT(1H0,'NR',11X,'EXPONENTS',20X,'PHI',12X,'JTE')              STEP0004
      K2=2*K+1                                                          STEP0006
      DO 14 I=1,K                                                       STEP0063
   14 B(I)=V(I)                                                         STEP0066
      DD=G                                                              STEP0068
      CALL LIN                                                          STEP0070
      AKLE=PHI                                                          STEP0080
      JTE=1                                                             STEP0085
      IR=0                                                              STEP0091
      WRITE (2,9)                                                       STEP0090
      DO 17 I=K1,4                                                      STEP0092
   17 B(I)=0.                                                           STEP0093
      WRITE(2,8)IR,(B(I),I=1,3),PHI,JTE                                 STEP0096
      DO 7 IR=1,IRMAX                                                   STEP0100
      JTEV=JTE                                                          STEP0150
      JTE=1                                                             STEP0200
      DO 1 J=1,K2                                                       STEP0210
      DO 1 I=1,K                                                        STEP0220
    1 BB(I,J)=B(I)                                                      STEP0230
      DO 23 J=2,K2                                                      STEP0260
      I=J/2                                                             STEP0270
      INT=1                                                             STEP0280
      IF(2*I.EQ.J) INT=-1                                               STEP0290
   23 BB(I,J)=BB(I,J)+INT*DD*D(I)                                       STEP0300
      PHIT=AKLE                                                         STEP0315
      DO 3 J=2,K2                                                       STEP0320
      IF ((JTEV/2*4-JTEV+1).EQ.J) GOTO 3                                STEP0325
      DO 4 I=1,K                                                        STEP0330
    4 B(I)=BB(I,J)                                                      STEP0340
      CALL LIN                                                          STEP0350
      IF (PHI.GE.PHIT) GOTO 3                                           STEP0351
      IF (PHI.GE.AKLE) GOTO 16                                          STEP0370
      JTE=J                                                             STEP0371
      AKLE=PHI                                                          STEP0380
   16 IF (J/2*2.EQ.J) J=J+1                                             STEP0381
    3 CONTINUE                                                          STEP0395
      DO 6 I=1,K                                                        STEP0430
      B(I)=BB(I,JTE)                                                    STEP0435
    6 BB(I,JTE)=BB(I,JTE)-BB(I,1)                                       STEP0440
      WRITE(2,8)IR,(B(I),I=1,3),AKLE,JTE                                STEP0445
      IF (JTE.NE.1)GOTO 21                                              STEP0454
   11 IF(DD.EQ.1) GOTO 12                                               STEP0455
      DD=DD/2                                                           STEP0460
      GOTO 7                                                            STEP0461
   21 DO 18 I=1,K                                                       STEP0462
   18 B(I)=B(I)+BB(I,JTE)                                               STEP0463
      CALL LIN                                                          STEP0464
      IF (PHI.LT.AKLE) GOTO 19                                          STEP0465
      DO 20 I=1,K                                                       STEP0466
   20 B(I)=B(I)-BB(I,JTE)                                               STEP0467
      GOTO 7                                                            STEP0468
   19 AKLE=PHI                                                          STEP0469
      WRITE(2,8)IR,(B(I),I=1,3),AKLE,JTE                                STEP0470
      GOTO 21                                                           STEP0471
    7 CONTINUE                                                          STEP0474
   12 CALL LIN                                                          STEP0478
      RETURN                                                            STEP0480
      END                                                               STEP0490
```

Fig. 4. Listing of STEP subroutine, algorithm C.

a step is tried (STEP0462–STEP0465). When the step is not succesful, the previous exponents are restored (STEP0466–STEP0467).

The STEP *subroutine according to* CACM *No.* 315

After evaluating the initial estimate (CACM 10–CACM 17) (Fig. 5) one step is made in each exponent (i.e. in one direction only), to determine the first derivative of the sum of least squares with respect to the exponents (CACM 18–CACM 25).

Then a step is tried in a direction determined by Newton's method (CACM 30–CACM 32).

When this step does not yield a sum of least squares which is low enough according to the damping criterion (CACM 38), or when the step would yield positive exponents (CACM 34), the size of the step is halved and the step is tested again. When a successful step has been taken, the whole procedure starts all over again until a certain minimum step-size is reached (CACM 45).

During each complete iteration one row of output is printed, containing the iteration number, exponents, sum of least squares, and step-size parameter BETA (CACM 37). When BETA = 1 the step-size is maximum (i.e. it is the original Newton step).

The LIN *subroutine*

The LIN subroutine (Fig. 6) calculates coefficients, constant and sum of least squares for a given set of exponents by simple matrix inversion.

```
      SUBROUTINE STEP                                          CACM   0
C                                                              CACM   1
C               VERSION ACCORDING TO CACM #315                 CACM   2
C               CANNOT YIELD POSITIVE B(I) DUE TO CACM0034     CACM   3
C                                                              CACM   4
      COMMON U(500,6),C(6),K,N,PHI,B(4),G,D(3),IRMAX,K1,V(4)   CACM   5
      DIMENSION DPHI(3),FPHI(3)                                CACM   6
    1 FORMAT(1H ,I3,3F10.5,F15.2,F15.8)                        CACM   7
    2 FORMAT(1H0)                                              CACM   8
      LAMBDA=0.2                                               CACM   9
   10 DO 11 I=1,3                                              CACM  10
   11 B(I)=V(I)                                                CACM  11
      DO 12 I=K1,4                                             CACM  12
   12 B(I)=0.                                                  CACM  13
   20 DO 80 IR=1,IRMAX                                         CACM  14
   30 CALL LIN                                                 CACM  15
      PHIC=PHI                                                 CACM  16
      WRITE(2,1)IR,(B(J),J=1,3),PHI                            CACM  17
      DO 31 I=1,K                                              CACM  18
      B(I)=B(I)+D(I)                                           CACM  19
      CALL LIN                                                 CACM  20
      FPHI(I)=PHI                                              CACM  21
      WRITE(2,1)IR,(B(J),J=1,3),PHI                            CACM  22
      B(I)=B(I)-D(I)                                           CACM  23
   31 CONTINUE                                                 CACM  25
   40 DO 41 I=1,K                                              CACM  26
      DPHI(I)=(FPHI(I)-PHIC)/D(I)                              CACM  27
      IF(ABS(DPHI(I)).LT..01) GOTO 120                         CACM  28
   41 CONTINUE                                                 CACM  29
   50 BETA=1                                                   CACM  30
   60 DO 61 I=1,K                                              CACM  31
   61 B(I)=B(I)-BETA*PHIC/DPHI(I)                              CACM  32
      DO 65 I=1,K                                              CACM  33
      IF(B(I).GE.0) GOTO 90                                    CACM  34
   65 CONTINUE                                                 CACM  35
      CALL LIN                                                 CACM  36
      WRITE(2,1)IR,(B(J),J=1,3),PHI,BETA                       CACM  37
   70 IF (PHI.GT.(1-LAMBDA*BETA)*PHIC) GOTO 90                 CACM  38
      WRITE(2,2)                                               CACM  39
      GOTO 80                                                  CACM  40
   90 DO 91 I=1,K                                              CACM  41
   91 B(I)=B(I)+BETA*PHIC/DPHI(I)                              CACM  42
  100 BETA=BETA/2                                              CACM  44
  110 IF(BETA.GE..00002) GOTO 60                               CACM  45
      GOTO 120                                                 CACM  46
   80 CONTINUE                                                 CACM  47
  120 RETURN                                                   CACM  48
      END                                                      CACM  49
```

Fig. 5. Listing of STEP subroutine, according to CACM No. 315 algorithm.

```
      SUBROUTINE LIN                                                  LIN00100
      COMMON U(500,6),C(6),K,N,PHI,B(4),G,D(3),IRMAX,K1,V(4)          LIN00200
      DIMENSION A(6,6)                                                LIN00300
      K2=K+2                                                          LIN00320
      K3=K+3                                                          LIN00330
      DO 1 J=1,K                                                      LIN00400
      I=1                                                             LIN00500
    3 ARG=B(J)*U(I,6)                                                 LIN00600
      IF (ARG.LT.-20.)  GOTO 2                                        LIN00700
      U(I,J)=EXP(+ARG)                                                LIN00800
      IF (I.GT.N) GOTO 1                                              LIN00900
      I=I+1                                                           LIN01000
      GOTO 3                                                          LIN01100
    2 U(I,J)=0.                                                       LIN01200
      IF (I.GT.N) GOTO 1                                              LIN01300
      I=I+1                                                           LIN01400
      GOTO 2                                                          LIN01500
    1 CONTINUE                                                        LIN01600
      DO 13 I=1,N                                                     LIN01700
      U(I,K2)=U(I,5)                                                  LIN01750
   13 U(I,K1)=1.                                                      LIN01800
      DO 4 L=1,K2                                                     LIN01900
      SOM=0.                                                          LIN02000
      DO 5 I=1,N                                                      LIN02100
    5 SOM=SOM+U(I,L)                                                  LIN02200
      A(1,L)=SOM                                                      LIN02300
      DO 4 M=1,K                                                      LIN02400
      M1=M+1                                                          LIN02500
      SOM=0.                                                          LIN02600
      DO 7 I=1,N                                                      LIN02700
    7 SOM=SOM+U(I,M)*U(I,L)                                           LIN02800
    4 A(M1,L)=SOM                                                     LIN02900
      DO 8 I=2,K1                                                     LIN03000
      DO 8 M=I,K1                                                     LIN03100
      DO 8 L=I,K2                                                     LIN03200
    8 A(M,L)=A(M,I-1)*A(I-1,L)-A(I-1,I-1)*A(M,L)                      LIN03300
      C(K1)=A(K1,K2)/A(K1,K1)                                         LIN03305
      DO 9 I1=2,K1                                                    LIN03310
      I=K2-I1                                                         LIN03320
      C(I)=A(I,K2)                                                    LIN03330
      DO 10 J1=2,I1                                                   LIN03340
      J=K3-J1                                                         LIN03350
   10 C(I)=C(I)-A(I,J)*C(J)                                           LIN03360
    9 C(I)=C(I)/A(I,I)                                                LIN03370
      SOM=0.                                                          LIN04000
      DO 11 I=1,N                                                     LIN04100
      U(I,4)=C(K1)                                                    LIN04110
      DO 12 J=1,K                                                     LIN04120
   12 U(I,4)=U(I,4)+C(J)*U(I,J)                                       LIN04130
   11 SOM=SOM+(U(I,4)-U(I,5))*(U(I,4)-U(I,5))                         LIN04600
      PHI=SOM                                                         LIN04700
      RETURN                                                          LIN04800
      END                                                             LIN04900
```

Fig. 6. Listing of LIN subroutine.

The matrix is formed in the statements LIN00100 up to and including LIN02900. A test had to be built in, to prevent underflow in the exponent subroutine (LIN00700). The matrix inversion is implemented by transforming the matrix into an upper-triangle matrix (LIN03000–LIN03300) and successive computation of the roots (LIN03305–LIN03370). Then the fitted function is generated (LIN04100–LIN04130) and the sum of least squares PHI is determined (LIN04600–LIN04700).

RESULTS

Test set-up

In order to test the algorithm under circumstances which approached the real measuring circumstances as closely as possible, a hardware exponential generator was built. It could generate a sum of three exponential terms and a constant, with or without additional white noise. The generator was sampled at constant intervals at a rate of 1 sample/sec by the computer for 1000 sec, and the samples were put in an array for digestion by subroutines STEP and LIN.

For economic reasons only 400 samples were used in the fitting procedure, viz. the first 200 samples and 200 equidistant samples chosen from the other 800. The trigger

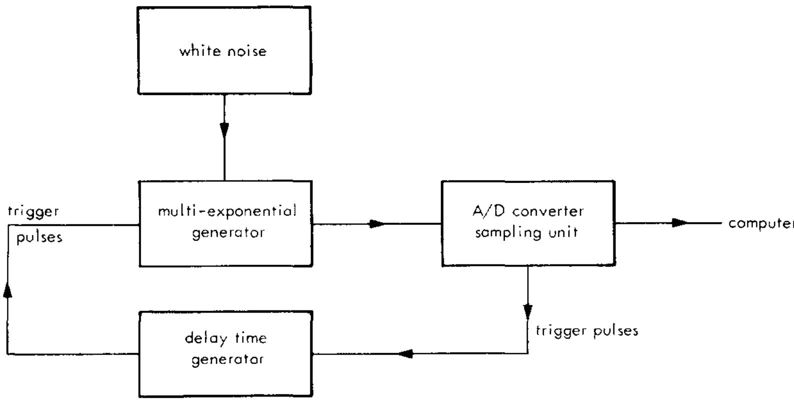


Fig. 7. Block diagram of test set-up.

pulses for the sampling of the A/D converter were fed back to the exponential generator in order to trigger the start of the exponential signal generation (see Fig. 7). This permitted investigation of the influence of the position of the first sample relative to the top of the signal (which is often random in practical measurements). The three exponential terms could also be generated separately in order to get a good estimate of the "real" values of the parameters.

In designing the generator, we chose the relaxation constants from the region where the relaxation constants are found in real experiments on bladder-wall strips [6]. In all cases the initial estimates of -0.40; -0.040 and -0.0040, were used with the minimum step-sizes 0.01; 0.001 and 0.0001, and a multiplication factor of 8. The coefficients and constant of the model were about equal. The amount of white noise added could be varied, and the frequency characteristic of the noise was flat down to the frequency corresponding to the fastest relaxation constant. To save time, "time scaling" was used, i.e. the experiments on the model generator were carried out ten times as fast as usual. Of course this scaling was incorporated in the computing of the results. Finally, the delay-time generator was adjustable between 0.1 Δt and 0.9 Δt where Δt is the sample time (1 sec). Two generated curves with and without added noise are presented in Figs. 8 and 9 respectively.

Influence of the position of the first sample

By generating only one exponential term, and varying the delay time we found that only the fastest exponential yielded a relaxation constant that depended on the delay time.

The relaxation constant found varied between 0.35 and 0.45 sec^{-1}, when the delay time varied between 0.1 Δt and 0.9 Δt (Table 1). This dependence can be easily understood if we note that the signal generated has, naturally enough, a rounded top. Now as a consequence of the trigger configuration, the computer has to start sampling before the exponential generator starts; so some samples have to be removed. The *real* start of the signal is determined by the highest sample in the series. Now let us consider the situation shown in Fig. 10. In this situation the first sample will be chosen as the highest, which will result in too low a relaxation constant. If this hypothesis reflects the real situation, then the real relaxation constant must be the highest one, and this one should give the lowest sum of least squares. This was found to be the case (see Table 1). Furthermore, the actual shape of the curve can be reconstructed by carefully blending the samples from all experiments with a variable delay time to yield a curve sampled at ten times the actual rate. This method also provided verification of the hypothesis. Finally it should be possible to eliminate the dependence on the delay time by simply removing the first sample (if enough samples are available, this should have no consequences for the relaxation constant); and indeed a relaxation constant of 0.45

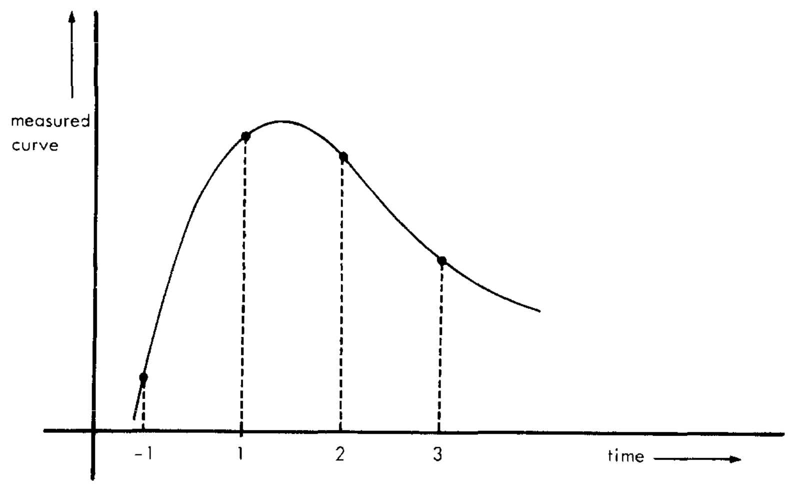


Fig. 10. Hypothetical sample distribution on generated curve.

sec^{-1} was always found if the first sample was removed. Now, though the two "slower" relaxation constants did *not* vary as a function of delay time when measured separately (because here the falling edge of the exponential function is very flat compared to the rising edge, so the situation of Fig. 10 is highly unlikely to occur) a dependence was found when the three exponentials were measured at the same time. The second relaxation constant then varied between 0.027 and 0.029 sec^{-1} and the third between 0.0028 and 0.0029 sec^{-1}. This must be due to correlation between the relaxation constants.

In the following tests, the delay time was fixed at the value which yielded the smallest sum of least squares when fitting three exponentials at the same time. In real experiments, where triggering of the system under measurement might be not possible, artefacts of the type described above can be avoided by picking as the first sample not the highest, but the one with the lowest first derivative.

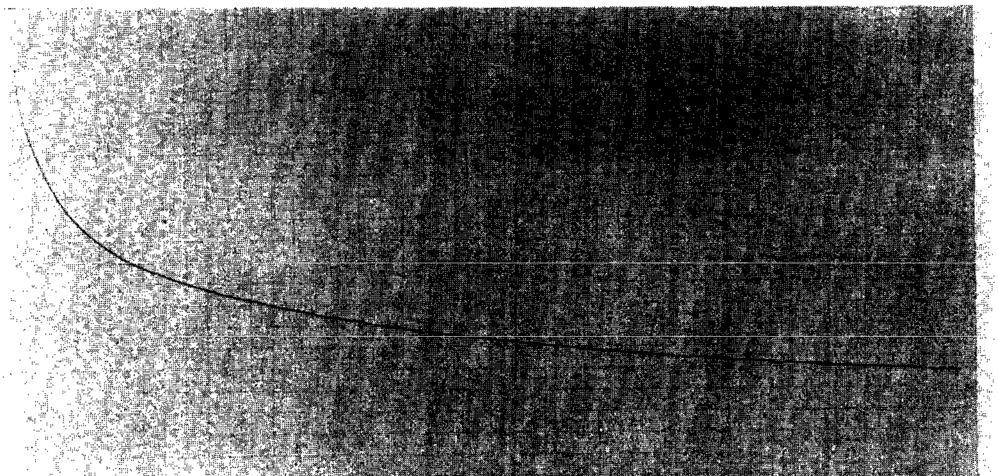
Influence of noise

A series of measurements was performed to test the influence of noise on the accuracy. The exponentials were first measured separately, which should yield very accurate values of the parameters. In fact, (see Table 2) no S.D. could be determined for the exponents because they were all exactly equal.

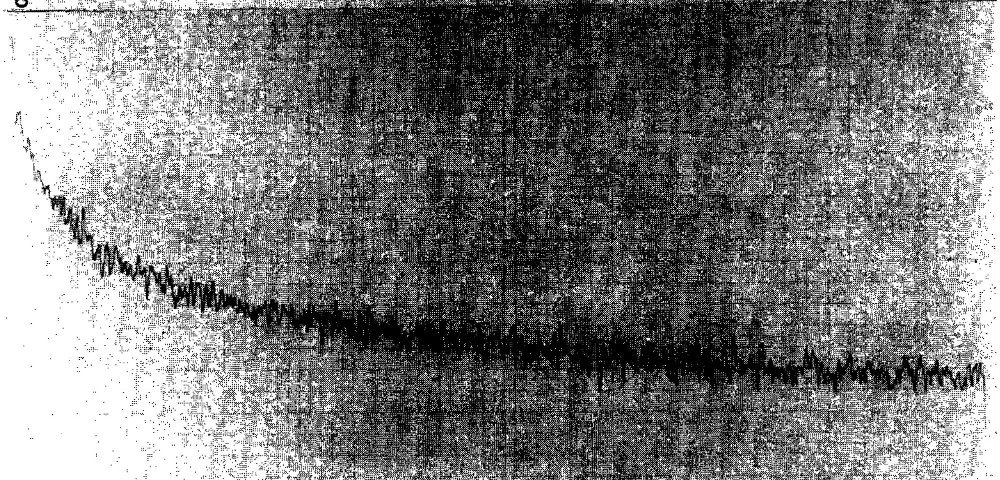
It should be borne in mind in this connection that the exponents are only determined with a limited accuracy equal to the step-size. For instance the step-size for the fastest exponent was 0.01. The result obtained (0.45) then means that an exponent of 0.44 or 0.46 would yield a higher sum of least squares. Since the value found in 11 measurements was always 0.45, no S.D. could be calculated. The coefficients (Table 3) were computed by solving a set of linear equations, which means that they can be determined to as many digits as desired; standard deviations could therefore be given for

Table 1. Influence of delay time on fastest exponent

Delay time	Relaxation constant	Sum of least squares
0.1 Δt	0.44	98
0.2 Δt	0.44	16
0.3 Δt	0.45	6
0.4 Δt	0.45	5
0.5 Δt	0.35	1599
0.6 Δt	0.37	967
0.7 Δt	0.39	508
0.8 Δt	0.40	290
0.9 Δt	0.41	251



8



9

Fig. 8. Three-exponential curve generated without noise.

Fig. 9. Three-exponential curve generated with noise ($=20$).

all these values, except for the constant a_0 which could be determined by simply switching off all exponentials and reading the value from the sampling interface display. It may be noted that the number of measurements though large enough varied quite considerably from run to run. This is due to the fact that the test set-up was fully automatic apart from switching on and off, so if the operator had a longer coffee break or was called to the telephone before the end of the run, more measurements were taken. The average sum of least squares can be split up into two components representing (a) the lack of fit between the signal and the model, and (b) the noise added to the signal.

Especially for the fastest exponent which was measured separately the sum of the least squares is very small, which is easily understood since only a few experimental points really contribute to the sum of the least squares here.

The relaxation constants could be determined with reasonable accuracy, as could the coefficients and constants (see Table 3). Note that the coefficient of the fastest exponent is smaller than the others, which must reduce the accuracy with which the fastest exponent can be determined. Furthermore the systematic error in the constant a_0 is equal to that in a_3, which means that a part of the slowest exponential must have been interpreted as a constant level.

Next, noise was added to the signal at different levels. The noise level could be checked from its contribution to the sum of least squares:

$$\phi\ (\text{noise}) = \phi\ (\text{total}) - \phi\ (\text{systematic}) = N \times \sigma^2_{\text{noise}}$$

so:

$$\sigma_{\text{noise}} = 5 \text{ should yield: } \phi = 10150$$
$$\sigma_{\text{noise}} = 10 \text{ should yield: } \phi = 40150$$
$$\sigma_{\text{noise}} = 20 \text{ should yield: } \phi = 160150.$$

This is in good agreement with the measured values. The results of the fittings with noise can be seen in Tables 2 and 3. It may be clearly seen that the fastest exponent is influenced much more than the slowest, as would be expected since fewer experimental points are available in practice in the former case.

In all cases, the minimum was found. When the "real" parameters were used as initial estimates, they yielded a higher sum of least squares than that found at the actual minimum.

It must therefore be concluded, that all errors shown in Tables 2 and 3 are due to the difficulty of separating a signal into exponentials. The stepwise approximation method always yielded the real minimum, which did not always agree with "real" values of the parameter. It is therefore *not* possible to obtain better results by *any other* least

Table 2. Influence of noise on exponents

Parameter	Value (s^{-1})	S.D. (%)	Systematic error (%)	Details of determination	Number of measurements	Average sum of least squares
γ_1	0.45	—	—	separately	11	5
γ_1	0.435	2%	3%	$\sigma_{\text{noise}} = 0$	12	150
γ_1	0.37	10%	18%	$\sigma_{\text{noise}} = 5$	13	10 843
γ_1	0.34	13%	24%	$\sigma_{\text{noise}} = 10$	22	41 540
γ_1	0.41	37%	9%	$\sigma_{\text{noise}} = 20$	31	160 674
γ_2	0.026	0	—	separately	15	2 100
γ_2	0.029	2%	12%	$\sigma_{\text{noise}} = 0$	12	150
γ_2	0.028	4%	8%	$\sigma_{\text{noise}} = 5$	13	10 843
γ_2	0.028	9%	8%	$\sigma_{\text{noise}} = 10$	22	41 540
γ_2	0.028	12%	8%	$\sigma_{\text{noise}} = 20$	31	160 674
γ_3	0.0029	0	—	separately	12	540
γ_3	0.0029	2%	0	$\sigma_{\text{noise}} = 0$	12	150
γ_3	0.0029	3%	0	$\sigma_{\text{noise}} = 5$	13	10 843
γ_3	0.0028	7%	3%	$\sigma_{\text{noise}} = 10$	22	41 540
γ_3	0.0029	10%	0	$\sigma_{\text{noise}} = 20$	31	160 674

Table 3. Influence of noise on coefficients and constant

Parameter	Value	S.D.	Systematic error	Details of determination	Number of measurements	Average sum of least squares
a_0	260	—	—	separately	—	—
a_0	276	0.4%	6%	three exponentials $\sigma_{noise} = 0$	12	150
a_0	274	0.8%	5%	$\sigma_{noise} = 5$	13	10 843
a_0	272	2%	5%	$\sigma_{noise} = 10$	22	41 540
a_0	271	4%	7%	$\sigma_{noise} = 20$	31	160 674
a_1	165	0.04%	—	separately	11	5
a_1	169	0.6%	2%	three exponentials	12	150
				$\sigma_{noise} = 0$	12	150
a_1	166	6%	0.6%	$\sigma_{noise} = 5$	13	10 843
a_1	157	9%	5%	$\sigma_{noise} = 10$	22	41 540
a_1	174	10%	5%	$\sigma_{noise} = 20$	31	160 674
a_2	269	1%	—	separately	15	2 100
a_2	268	0.2%	0.4%	three exponentials $\sigma_{noise} = 0$	12	150
a_2	265	2%	1%	$\sigma_{noise} = 5$	13	10 843
a_2	266	2%	1%	$\sigma_{noise} = 10$	22	41 540
a_2	265	5%	1%	$\sigma_{noise} = 20$	31	160 674
a_3	275	0.3%	—	separately	12	540
				three exponentials		
a_3	291	0.3%	6%	$\sigma_{noise} = 0$	12	150
a_3	290	1%	5%	$\sigma_{noise} = 5$	13	10 843
a_3	290	2%	5%	$\sigma_{noise} = 10$	22	41 540
a_3	291	3%	6%	$\sigma_{noise} = 20$	31	160 674

squares method. The only possible way of obtaining a lower sum of least squares would be to determine the relaxation constants to a higher number of digits. However, there seems to be little point in this in view of the reasonably low S.D. obtained. Despite all the difficulties involved in separating signals into exponentials, we thus see that this is possible with a very reasonable accuracy under the given circumstances.

Rate of convergence of algorithm

The program was run on a Texas Instruments 980B minicomputer, with hardware multiply/divide, and software floating-point processing. It was found that the evaluation of one point in the parameter space (one call to subroutine LIN) when fitting three exponentials to 400 experimentals points took 10 sec. (Experimental runs on a PDP 9 and Nova 2/10, both with hardware multiply/divide and software floating point also yielded 10 sec in both cases.) The rate of convergence of algorithms can be compared by determining the number of parameter-point evaluations needed to reach the minimum.

For a three-exponential fit, one iteration using algorithm A involves 7 evaluations (central point + 2 steps in each of 3 directions). Using algorithm B involves an average of 3.75 evaluations (viz. 7-the central point-the previous point-$\frac{1}{4}$ × the steps in the other directions if we assume that in half of the cases the first. positive step in a certain direction yields a fall in ϕ, so that the step in the opposite direction can be omitted) while using C involves either 4.75 evaluations (as for B plus the extra step) or 1 evaluation (when only the extra step is needed). It was found that when the initial estimates were not too bad, the optional extra step of algorithm C was hardly ever used, so algorithm B must be considered as the fastest. Convergence from the initial estimates (−0.40, −0.040, and −0.0040) to the results of Tables 2 and 3 (without noise) took an average of 22 iterations, using the minimum step-size 0.01, 0.001 and 0.0001 and a multiplication factor of 8. With algorithm B this means that 83 parameter points had to be evaluated which took 830 sec or *ca.* 14 min. This may seem rather a lot, but it may be mentioned for the sake of comparison that the advanced iteration scheme according to CACM No. 315 involves the evaluation of 102 parameter points for convergence from the same initial estimates (Table 4).

Table 4. Parameters determined by an algorithm according to CACM No. 315

Parameter	Value	S.D.	Systematic error	Details of determination	Number of measurements	Sum of least squares
γ_1	0.44	10%	2%	$\sigma_{noise} = 0$	11	462
γ_2	0.029	4%	12%	$\sigma_{noise} = 0$	11	462
γ_3	0.0030	3%	3%	$\sigma_{noise} = 0$	11	462
a_0	290	1%	11%	$\sigma_{noise} = 0$	11	462
a_1	174	2%	5%	$\sigma_{noise} = 0$	11	462
a_2	275	2%	2%	$\sigma_{noise} = 0$	11	462
a_3	303	1%	10%	$\sigma_{noise} = 0$	11	462

(It should be remembered in this connection that the derivative of the sum of least squares with respect to the parameters was estimated by making small steps in the axial direction of all parameters, which involves at least k evaluations per iteration, where k is the number of exponentials). Furthermore, the algorithm of CACM No. 315 did not converge to the real minimum (average sum of least squares = 462 for a three-exponential fit to a curve without noise, while the constant step approximation method yields an average sum of least squares of 150; this means that iteration stopped too early).

In all cases one of the three steps used to determine the first derivative of the sum of least squares actually yielded a lower sum of least squares than finally reached. This could not be improved by varying the damping factor λ. The iteration seemed to be almost totally insensitive to variations in λ: varying this parameter between 0.1 and 0.99 did not influence the results, though $\lambda = 1$ did yield a significantly worse result.

DISCUSSION

It is generally acknowledged that fitting exponentials to measured data is a very troublesome task. Two kinds of difficulties can be distinguished:

(1) The minimum in the quality function (in this case the sum of least squares) may be hard to detect.

(2) The minimum may not agree with the "real" values of the parameters. It will be clear that difficulties of this kind can be detected (by testing with models as described in the preceding Section) but cannot be resolved, unless we use another kind of quality function. However difficulties of the first kind can be resolved for if a minimum however shallow exists, it should be possible to work out refined methods to detect it.

Our constant-step approximation method was designed on the assumption that the minimum in the quality function does agree with the real values of the parameters but may be hard to detect. If follows that this method has the following features:

(1) The procedure is very simple; this means that little can go wrong

(2) The procedure yields a clear insight into the form of the minimum; i.e. in the intractability of the given set of data.

(3) With a flat minimum, the procedure ensures that steps are continued until the quality function starts rising again, thus making sure that we don't stop before the minimum is reached.

Of course our method does involve a lot of computation, but with a minicomputer at one's disposal one can afford himself to explore a considerable part of the parameter space.

Besides, a method which would be expected to be much more efficient such as the CACM No. 315 actually turns out to need more time to converge to a worse approximation to the minimum, when the initial estimates supplied are not too bad.

Concerning the accuracy which can be reached we may state that when the relaxation constants involved are about one order of magnitude apart, they can be determined with satisfying accuracy, even if a lot of white noise is added to the signal.

SUMMARY

Description of a FORTRAN IV computer program which fits up to three exponential terms and a constant to experimental data according to a least squares criterion.

The search is performed by taking fixed steps in a number of directions in the parameter space by varying the exponents, and choosing the best direction. The coefficients and the constant are computed by a classical least squares method. The program was tested on hardware generated three-exponential curves with additional white noise. The accuracy obtained was satisfactory. The program uses a considerable amount of CPU time, though less than a program according to the CACM No. 315 algorithm which was tried for comparison.

This more advanced algorithm applies a damped Newton iteration but it turned out to be unable to detect the minimum in the least squares function as accurately as the stepwise program described. Finally, the program is short enough to be run on a minicomputer.

REFERENCES

1. B. L. R. A. Coolsaet, W. A. van Duyl, R. van Mastrigt and A. J. van der Zwart, *Urology* **2**, 255–257 (1973).
2. A. Kondo, J. G. Susset and J. Lefaivre, *Invest. Uro.* **10**, 154–163 (1972).
3. A. J. van der Zwart, Report M 106, Technical University of Delft, November (1973).
4. C. Laczos, *Applied Analysis*. Prentice-Hall, Englewood Cliffs, NJ (1956).
5. B. L. R. A. Coolsaet, W. A. van Duyl, R. van Mastrigt and J. W. Schouten, *Invest. Urol.* **12**, (1975).
6. B. L. R. A. Coolsaet, R. van Mastrigt, W. A. van Duyl and R. E. F. Huygen, *Invest. Urol.* **13**, 435–440 (1976).

LIST OF SYMBOLS

a_0	constant in exponential model
a_i	coefficient of i^{th} exponential
i	subscript
j	subscript
k	number of exponential terms used
N	number of experimental points
x_i	independent variable of set of experimental points
y_i	dependent variable of set of experimental points
γ_i	relaxation constant in i^{th} exponential term (s^{-1})
Δt	sampling time (s)
σ_{noise}	S.D. of noise
ϕ	sum of least squares.

LIST OF NAMES OF VARIABLES IN THE PROGRAM

AKLE	smallest sum of least squares
$B(I)$	relaxation constants of I^{th} exponential term
BETA	step-size in CACM No. 315
$C(I)$	coefficient of I^{th} exponential
$C(K1)$	constant ($K1 = K + 1$)
$D(1)$, $D(2)$, $D(3)$	final step-sizes
G	multiplication factor for step-size
IRMAX	maximum number of iterations
JTE	direction number
JTEV	direction number of previous iteration
K	number of exponentials fitted
LAMBDA	damping factor in CACM No. 315
N	number of experimental points
PHI	sum of least squares
PHIT	sum of least squares of previous iteration.
$U(I,4)$	array with fitted function
$U(I,5)$	array with Y data (dependent variable)
$U(I,6)$	array with X data (independent variable)
$V(I)$	initial estimate of I^{th} relaxation constant (sec^{-1}).

About the Author—ROBERT VAN MASTRIGT was born in Rotterdam, on 29 July 1950. He received his Masters Degree in Applied Physics in 1972 from the Technical University of Delft. In the same year he joined a group consisting of B. L. R. A. Coolsaet, urologist and W. A. van Duyl, engineer, who were performing interdisciplinary research on the physical properties of the urinary tract at the Erasmus University Rotterdam. A number of articles have already been published on the visco-elastic properties of the urinary bladder. At the moment the author is preparing a thesis on the passive properties of the urinary bladder in the collection phase.