Behavior Research Methods & Instrumentation
1975, Vol. 7 (4), 369-376

COMPUTER TECHNOLOGY

A random search algorithm for laboratory computers

RENWICK E. CURRY
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

The small laboratory computer is ideal for experimental control and data acquisition. Postexperimental data processing is many times performed on large computers because of the availability of sophisticated programs, but costs and data compatibility are negative factors. Parameter optimization, which subsumes curve fitting, model fitting, parameter estimation, least squares, etc., can be accomplished on the small computer and offers ease of programming, data compatability, and low cost, as attractive features. A previously proposed random search algorithm ("random creep") was found to be very slow in convergence. We present a new method (the "random leap" algorithm) which starts in a global search mode and automatically adjusts step size to speed convergence. A FORTRAN executive program for the random leap algorithm is presented which calls a user supplied function subroutine. An example of a function subroutine is given which calculates maximum likelihood estimates of receiver operating characteristic parameters from binary response data. Other applications in parameter estimation, generalized least squares, and matrix inversion are discussed.

For many investigators involved in behavioral research, the small laboratory computer is viewed primarily as an experiment monitor, controller, and data acquisition device. The small computer is ideal for this purpose because of its low initial cost and because it is dedicated to the laboratory in which it resides.

The decision is not as clear when considering whether or not to perform postexperimental data processing on the small computer. In favor of the small computer are data compatibility (no tape-to-tape or tape-to-card conversions), low operating cost, and availability. Furthermore, there are many programs designed for these applications (e.g., the DECUS library).

The advantages of processing the experimental data on a large computer are the increases in flexibility, scope, and sophistication of processing techniques attendant with the increase in memory and computational speed. There are many well-documented programs in existence which take advantage of these attributes (e.g., the UCLA BMD series).

A strong case can be made for the small computer in one area of "sophisticated" data processing, and it has received only a modest amount of attention. Parameter optimization, which subsumes curve fitting, model fitting, parameter estimation, least squares, polynomial root finding, etc., is a task which can be performed on the large computer as well as the small. Many algorithms have been proposed (e.g., Fletcher & Powell, 1963) and several of these are available in coded form (e.g., IBM Scientific Subroutine Package). These algorithms typically require gradient calculations, some require matrix inversions, and all would quickly overwhelm the capabilities of a small computer. As with most algorithms, they only attempt fo find one local extremum.

Other algorithms utilize a direct search process and are more readily implemented on a small computer. They are conceptually the most simple: the objective function or cost function $f(\cdot)$ is evaluated at a point in parameter space $\hat{x}$; a trial point $\tilde{x} = \hat{x} + \Delta x$ is selected, and if $f(\tilde{x})$ is an improvement over $f(\hat{x})$, $\hat{x}$ is replaced by $\tilde{x}$. Thus, the best solution is always retained and relinquished only when a better one is found.

Algorithms of the direct search type are differentiated by the manner in which the trial value (or Δx) is chosen. Examples of direct search algorithms are Chandler's (Note 1) STEPIT and Hooke and Jeeves' procedure (1961); these and other methods are compared in Dorfman, Beavers, & Saslow (1973). The performance of these algorithms, as measured by the conventional yardsticks of computing time or the number of times the cost function is evaluated, is usually (but not always) inferior to the more sophisticated approaches. For the small computer, however, time is *not* money, and when the further advantages of less programming effort (e.g., no gradients to be computed) and data compatibility are considered, the direct search algorithms on a small computer become very attractive.

One final feature, and some would say that it is the

This research was sponsored by NASA Grant NGR 22-009-733, Man-Machine Integration Branch, NASA-AMES Research Center.

most important, is that these techniques can more easily search for global, not just local extrema, since the logic for global search is readily incorporated in the direct search logic.

In this paper, we present a direct search program for function minimization which is intended especially for use on small computers. Because of this goal, the primary consideration is small program size rather than speed of computation, number of function evaluations, or computation costs. The program's genesis is the "random creep" algorithm originally proposed by Faureau and Franks (1958) and discussed most recently by Bekey and Ung (1974). After describing the algorithm, we propose a new algorithm (the "random leap") which offers substantially better convergence speed. A FORTRAN program to carry out the random leap procedure is given, and an example of maximum likelihood estimation of receiver operating characteristic parameters is presented. Applications to generalized least squares and matrix inversions are also discussed.

RANDOM SEARCH ALGORITHMS FOR GLOBAL EXTREMA

The Random Creep Method

The random creep algorithm (Bekey & Ung, 1974; Faureau & Franks, 1958) derives its name from the manner in which the trial values of the parameter vector increment Δx are derived: (1) At each stage of the iteration, each of the n elements of the currently optimal parameter vector $\hat{x}$ is perturbed by a zero-mean Gaussian random number of specified standard deviation:

$$\Delta x_j^i = \epsilon_j^i \qquad j = 1, \ldots, n$$

$$\epsilon_j = N(0, \sigma_j^2)$$

where Δx_j^i is the perturbation to the jth element of x on the ith trial, and the ϵ_j are independent zero-mean Gaussian variables. (2) If the trial vector $\tilde{x} = \hat{x} + \Delta x$ results in an improvement, $\hat{x}$ is replaced by $\tilde{x}$. If not, a new Δx is chosen according to the above equation. (3) If the iteration fails to improve after a specified number of consecutive trials, all standard deviations are increased by the same factor

$$\sigma_j = \alpha \sigma_j \qquad j = 1, \ldots, n$$

where $\alpha > 1$. This action is based on the assumption that a local minimum has been reached. (4) The process terminates after either (a) the total number of iterations reaches a predetermined value, or (b) the standard deviations have been increased a specified number of times.

The major advantage of this approach in small computer applications is the relatively small amount of storage space required for the search algorithm. There is

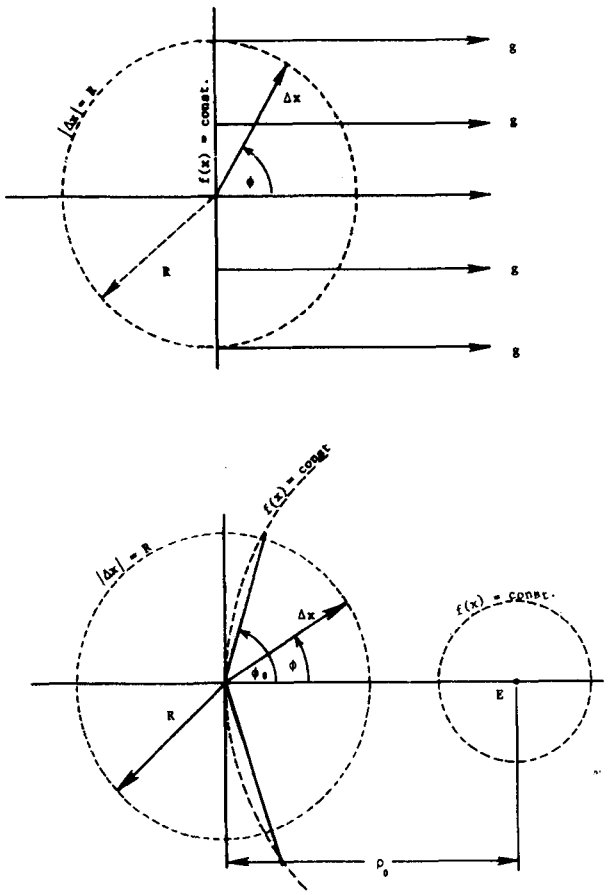

Figure 1. Geometry of possible improvement with random steps (Δx) of fixed size (R) in two-parameter space. (a) Far from the extremum: the gradient vectors (g) are parallel, and an improvement is possible only if ϕ (the angle between Δx and g) is $-\pi/2 < \phi < \pi/2$, which occurs with the probability .5. (b) Close to a spherical extremum (E) at distance ρ_0: contours of constant criterion f(x) are concentric circles centered at E. Improvement is possible only if $-\phi_0 < \phi < \phi_0 < \pi/2$ with probability < .5. When the step size is greater than twice the distance to the extremum ($R > 2\rho_0$), no improvement is possible.

no guarantee that this algorithm will converge to the global extremum (only an exhaustive search will do that), but experience has shown this to be an effective method of finding local and global extrema (Bekey & Ung, 1974).

The Random Leap Method

Improving convergence rate. The main disadvantage of the random creep method, as it is described in the previous section, is the relatively slow rate of convergence because only the minimum step size is used. We experimented with modifications to rectify this problem; for example, if Δx had been an improvement on the previous trial, it was then used in succeeding trials until no further improvement was obtained. These modifications gave a minimal amount of improvement.

It was about this time that the analytical work of Rastrigan (1963) came to our attention; he analyzed a random search procedure where the direction of the

random search vector is uniformly distributed over the hypersphere, but the step size is fixed. The areas of interest are shown in Figure 1 for a two-parameter case: Δx is the random increment in the parameter vector and has a fixed magnitude or step size of R; ϕ is the angle between the gradient vector g and the random increment Δx. In Figure 1a, the currently optimal value of x is far from the extremum resulting in (nearly) parallel gradients throughout the trial region. The probability of an improvement is ½ under these conditions.

Figure 1b shows the situation when the search procedure nears an extremum E which is assumed to be of a spherical nature—the contours of constant cost function are circles in this two-parameter case, but are hyperspheres in general. In this diagram, the step size is again denoted by R; ϕ is the angle of Δx from the gradient, ϕ_0 is the maximum angle for which an improvement can be made; and ρ_0 is the distance of the current point from the extremum. It can be seen that the probability of improvement is less than ½ under this condition, since there is a smaller region in which the random search vector will yield an improvement. In the extreme case, when the step size is twice the distance to the extremum, then all points of the function which would yield an improvement lie within the circle of the search vector and no improvement is possible. These factors are shown quantiatively in Figure 2, which

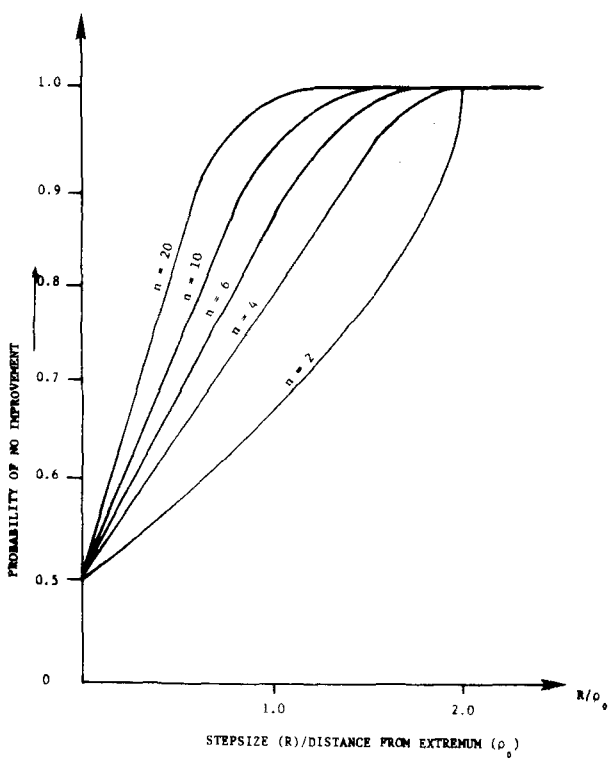

Figure 2. Probability of no improvement vs. step size (measured in distance to extremum) for spherical extremum in n-dimensional space.

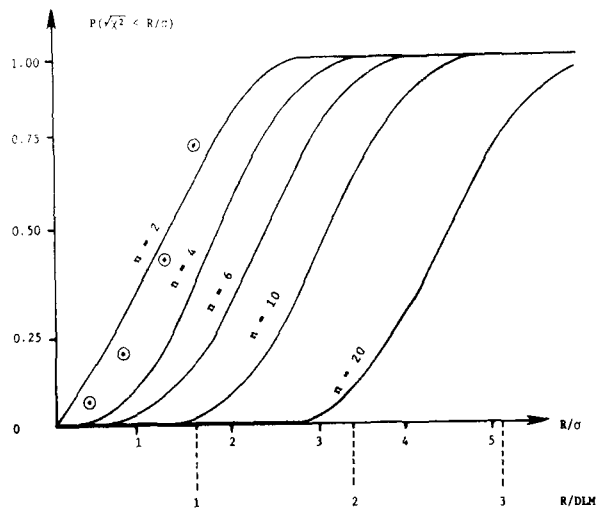

Figure 3. Approximate magnitude R of the random search increment Δx. Upper abscissa scale for R normalized by the standard deviation (Gaussian distribution); lower abscissa scale for R normalized by the half-limit of a uniform distribution (DLM) = $\sqrt{3\sigma}$). Circles show exact values for uniform distribution (n = 2).

demonstrates the probability of no improvement for fixed step size as a function of the step size, measured in distance from a spherical extremum, for various dimensions of the parameter vector. These curves were generated by numerically integrating the expressions given in Rastrigan (1963).

Figure 2 contains the information that is the basis for the random leap method. In essence, the step size is adjusted so that one always obtains a modest probability for improvement. When the step size is large, it is difficult to get an improvement, so the step size is halved. As the search procedure then comes closer to the extremum, it again becomes difficult to obtain an improvement and the step size is again halved.

When dealing, as we are, with a completely random increment in each coordinate direction, the step size is not fixed. In Figure 3, we show the cumulative distribution curves for the $\sqrt{\chi^2}$ distribution which is the magnitude of the random increment Δx when the increments are independent Gaussian variables with the same standard deviation. The algorithm we suggest uses uniformly distributed variables, so we have also shown, for later use, the abscissa in the scale of step size normalized with respect to the limits of the uniform distribution. The difference between the uniform and Gaussian distributions (in terms of the magnitude of the vector) becomes small for a large number of parameters; the circles in Figure 3 indicate the exact solutions for uniformly distributed variables for n = 2, and even for this low number of parameters, the deviations are of no practical significance.

Program outline. The overview of the operation begins with a call to the user-supplied function routine to

```
C******RANDOM LEAP PARAMETER SEARCH ROUTINE
      DIMENSION X(5),TMPX(5),DLM0(5),TMPLM(5)
C******SET IA=2**(P/2)-3,RNDMG=2**(P-1)-1 FOR P BIT MACHINE
      IA = 61
      RNDMG = 2047
      IRND = 1237
C******INITIALIZATION CALL TO GET PARAMETERS AND FIRST VALUE
    1 LFRST = 1
      TCST=CSTFN(TMPX,NX,LFRST,MXITR,MXFGS,MXEXP,MXFLS,MXCTR,LPRT,DLM0)
      LFRST = 2
      NFAIL = 0
      ITER = 0
C******START LOCAL SEARCH HERE
   30 MODE = 1
C******START GLOBAL SEARCH HERE (MODE SET TO 2 BELOW)
   35 NEXP = 0
      NCTR = 0
C******SET TEMPORARY DISTR. LIMITS AT INITIAL VALUE
      DO 40 I=1,NX
   40 TMPLM(I) = DLM0(I)
      GO TO (50,80),MODE
C******SAVE SUCCESSFUL TRY AND PRINT IF LPRNT=2
   50 CST = TCST
      DO 60 I=1,NX
   60 X(I) = TMPX(I)
      GO TO (75,70),LPRT
   70 WRITE (1,1040) ITER,MODE,NFAIL,NEXP,NCTR,CST,(X(I),I=1,NX)
1040  FORMAT (5I6,E15.5,EF10.4/(10X,6F10.4))
   75 NFAIL = 0
C******FIND AND TEST NEW TRIAL X AFTER CHECKING FOR MAX ITERATIONS
   80 ITER = ITER + 1
      IF (ITER - MXITR) 90,90,230
   90 DO 99 I=1,NX
      IRND = IA*IRND
      IF (IRND) 99,230,99
   99 TMPX(I) = X(I) + FLOAT(IRND)/RNDMG*TMPLM(I)
      TCST=CSTFN(TMPX,NX,LFRST,MXITR,MXFGS,MXEXP,MXFLS,MXCTR,LPRT,DLM0)
      IF (TCST - CST) 100,110,110
C******SAVE IMPROVED SOLUTION BUT FIRST SWITCH TO LOCAL IF IN GLOBAL
  100 GO TO (50,30),MODE
C******OTHERWISE COUNT SUCCESSIVE FAILURES, GO TO STEPSIZE LOGIC
  110 NFAIL = NFAIL + 1
      GO TO (120,170),MODE
C******STEPSIZE LOGIC FOR LOCAL SEARCH
C******TEST FOR MAX FAILURES, LOCAL SEARCH
  120 IF (NFAIL - MXFLS) 80,80,130
  130 NCTR = NCTR + 1
      IF (NCTR - MXCTR) 140,140,160
C******HALVE DISTRIBUTION LIMITS AND TRY NEW X
  140 NFAIL = 0
      DO 150 I=1,NX
  150 TMPLM(I) = TMPLM(I)*.5
      GO TO 80
C******END LOCAL SEARCH IF TOO MANY CONTRACTIONS (HALVINGS)
  160 WRITE (1,1100)
1100  FORMAT ('END LOCAL SEARCH')
      WRITE (1,1040) ITER,MODE,NFAIL,NEXP,NCTR,CST,(X(I),I=1,NX)
      MODE = 2
      GO TO 35
C******STEPSIZE LOGIC FOR GLOBAL SEARCH
C******TEST FOR MAX FAILURES, GLOBAL SEARCH
  170 IF (NFAIL - MXFGS) 80,80,180
  180 NEXP = NEXP + 1
      IF (NEXP - MXEXP) 190,190,230
C******EXPAND DISTRIBUTION LIMITS
  190 NFAIL = 0
      DO 200 I=1,NX
  200 TMPLM(I) = 1.3*TMPLM(I)
      GO TO 80
C******EXIT SEARCH LOGIC HERE AND START AGAIN
  230 WRITE (1,1060)
1060  FORMAT ('STOP')
      WRITE (1,1040) ITER,MODE,NFAIL,NEXP,NCTR,CST,(X(I),I=1,NX)
      GO TO 1
      END
```

Figure 4

supply the constants for the program operation and the initial guess at the parameter vector. From there, the operation proceeds as follows: (a) Uniformly distributed independent increments of each coordinate are chosen to calculate the new trial value of x.[1] Initially, the distribution limits of these random increments are chosen to be of moderate size relative to the entire search region, i.e., if the search region is $(-1,1)$ in each coordinate, then the limits of the uniform increments are on the order of $(-1,1)$. This allows a preliminary global search in the beginning phases. (b) When the trial values of x have failed to yield an improvement a specified number of times, it is assumed that the step size is too large relative to the distance to the extremum, and the distribution limits are halved. The search continues with these limits until the procedure fails to yield an improvement in the (same) consecutive number of trials. (c) After the distribution limits have been halved a specified number of times, it is assumed that the procedure has converged to a local minimum. The results are printed out, and the distribution limits are reset to their original, moderately sized, values. The global search is then initiated in a manner similar to the random creep method: if no improvement is reached after a specified number of times, the distribution limits are increased, and the search continued. Note, however, that the global search is initiated with the moderately sized distribution limits and not the minimum size, as is done in the random creep method. (d) The procedure is terminated whenever the total number of iterations exceeds a specified value or when the global search has not yielded an improvement after a specified number of expansions of the distribution limits.

The advantage of the random leap algorithm lies in its ability to perform a preliminary global search and gradually reduce the step size as the extremum is neared. There are cases, obviously, when the random creep method would be better, such as when the initial value of x is very close to the extremum.

Program description. A FORTRAN listing of the random leap algorithm is given in Figure 4, which executes the general procedure described above. To do this, it requires a user-supplied function subroutine CSTFN which calculates the value of the cost function; we have made provision for the user to enter the constants and parameters for the search routine for maximum flexibility. During the first call to the function routine CSTFN, the following transfer of variables obtains:

Variables Passed to CSTFN

LFRST has been set to the value of 1 to indicate that this is the first (initialization) call

Variables Returned from CSTFN

TMPX temporary value of X, in this case the initial value

NX the number of dimensions in the parameter vector x

MXITR maximum number of iterations

MXFGS maximum number of consecutive failures in the global search mode

MXEXP maximum number of expansions (increases of the distribution limits) in the global search mode

MXFLS maximum number of consecutive failures in the local search mode

MXCTR maximum number of contractions (halvings) of the distribution limits in the local search mode

LPRT returned as 2 to print (a) each time an improvement is made, (b) at the end of the local search, and (c) at the termination of the search; returned as 1 to print only at the end of the local search and at the end of the entire search

TCST the numerical value of the cost function
DLMØ the array containing the initial limits of the uniformly distributed random search increments. For example, ΔX(1) is uniformly distributed between (−DLMØ(1), DLMØ(1)).

For subsequent calls to the function routine during the normal course of operation, the following transfer of variables applies:

Variables Passed to CSTFN

TMPX the temporary or trial value of X
LFRST has been set to 2 to indicate that initialization is not required

Variables Returned from CSTFN

TCST the numerical value of the cost function

CHOOSING THE SEARCH PARAMETERS

In this section, we give some guidelines on how one determines the parameters of the search. Assume that the parameters have been scaled such that the solution is likely to lie within the unit hypercube (−1,1) in each coordinate. Let us pick a convergence criterion that (say) each value should be determined within at least .01 of the true value which minimizes the cost function. To find the minimum step size required, we arbitrarily fix the probability of no improvement which we would like to detect at (say) .7. Looking at Figure 2, we can then determine the step size relative to the distance from the extremum for various dimensions of the parameter vector. If we have four parameters, then a probability of .7 occurs with a step size that is roughly .6 times the distance from the extremum. Since we want the distance from the extremum to be no more than .01, the magnitude of the minimum step size must be .006. To find the minimum distribution limits corresponding to the minimum step size, we refer to Figure 3, where we see that for n = 4, the step size will be less than 1.8 times the distribution limit more than 95% of the time. Thus, the minimum limits of the uniform distribution should be ±.006/1.8 = ±.0033. The initial value of the distribution limits should be on the order of ±1, and because the limits are halved at each stage, we seek an integer M such that $1/2^M \simeq .0033$. Note that $2^8(.0033) = .768$, so we set DLMØ = .8 and MXCTR = 8.

The adaptive nature of the random leap program results from testing the hypothesis that the probability of failure is .7 or less. To reject this hypothesis at the .05 and .01 levels of significance, we would need 9 amd 13 successive failures, respectively. (six and nine successive failures are required for the same level of confidence when detecting probability of failure .6 or less, showing the desirability of working with probabilities of no improvement closer to .5 than 1.0.) Thus, MXFLS, Maximum Consecutive Failures, Local Search, should be in the range of 9 to 13, perhaps greater to account for contingencies.

The parameters for the global process (MXFGS and MXEXP) can really only be determined by experimentation in each particular situation. We found, in the cost function described below (which has two maxima) that 300 trials with a five-dimensional vector were more than adequate to find the global maximum when situated at the local maximum. The number of such trials must be increased substantially when going to a larger parameter vector to obtain similar densities of trial values within the parameter space.

APPLICATIONS

Maximum Likelihood Estimation

In this section, we describe an application of the random leap algorithm and the results of a Monte Carlo simulation. The first example uses a cost function of the form

$$f(x) = .59\exp(-Q_1(x)) + .41\exp(-Q_2(x)) \quad (1)$$

$$Q_1 = S(n)\sum_{i=1}^{n}(.5 - x_i)^2/(1.26)^{i-1} \quad (2a)$$

$$Q_2 = S(n)\sum_{i=1}^{n}(.5 + x_i)^2/(1.26)^{i-1} \quad (2b)$$

where S(n) is a scale factor depending only on the number of dimensions in the parameter vector and keeps the argument of the exponent function within reasonable bounds. This function was chosen for a detailed examination because of its two nearly equal modes. It corresponds to a maximum likelihood estimation problem in which the observations (.5, .5, . . .) come from either a distribution with mean x (prior probability .59) or −x (prior probability .41) and unequal variances.

Monte Carlo trials. One hundred Monte Carlo trials of the random creep and random leap operations were run using the cost function in Equation 1 on a PDP-12 computer with software multiply and divide routines. Approximate time for each iteration was about 1.5 sec. The initial conditions for these trials were uniformly distributed in the unit hypercube, and parameter vectors of dimensions 2 and 5 were used. Convergence was considered complete when the search routine first came within a sphere corresponding to an rms deviation in each component of .05.

With the two-dimensional parameter vector, the mean number of iterations-to-convergence for the random leap algorithm was 35, the median 36, and the maximum number of iterations required was 67. In all these cases, the fact that the search was started at the global level rather than the local level resulted in a convergence to the global maximum first. On the other hand, the random creep algorithm converged to the local

maximum first in many trials. It also failed to find the global maximum within the allowable maximum of 500 iterations on many of its trials, whereas the random leap algorithm never failed to converge to the global maximum.

In testing the random leap algorithm with the five-dimensional parameter vector, the mean number of iterations-to-convergence in 100 Monte Carlo trials was 194, with a maximum number of iterations of 365 and a minimum of 58. The distribution of the number of iterations had two modes: one, at 100 iterations, corresponded to those searches that went directly to the global maximum; the other mode, at 240 iterations, represented those searches that went to the local maximum first and then to the global maximum. Sixty of the 100 Monte Carlo trials converged to the global maximum first; this is a statistically significant difference from the expected value of 50 that one would expect with the random creep algorithm, and is due to the fact that the random leap algorithm starts off in the global search mode. The random creep algorithm was not run in this more difficult task because of its relatively poor performance on the easier two-parameter case.

Receiver Operating Characteristics

In the theory of signal detection (TSD) approach to psychophysics, the subject's response is divided into sensory and response bias components. One may administer a yes-no response procedure in several sessions, during each one of which a subject adopts a different strategy or criterion level (β), whereas the sensitivity of the signal (d') remains constant throughout (Green & Swets, 1966). Assuming that we have binary response data ("YES" or "NO") for M criterion levels, the TSD model uses the following expression for the probability of a "YES" response given noise (n) and signal(s) respectively

$$P(\text{"YES"}|\text{noise}) = 1 - \Phi(\beta_i) = \Phi(-\beta_i) \qquad (3)$$

$$i = 1,2,\ldots,M$$

$$P(\text{"YES"}|\text{signal}) = 1 - \Phi[(\beta_i - d')/\sigma_s] = \Phi[(d' - \beta_i)/\sigma_s] \qquad (4)$$

where β_i is the criterion level adopted in the ith session, d' is the sensitivity measured in noise standard deviation units, σ_s is the standard deviation of the signal distribution, measured in noise units and Φ is the Gaussian distribution function.

The maximum likelihood estimates for the parameters β_i, d', and σ_s are obtained by maximizing the likelihood function which is proportional to

$$\ell(x) \propto f(x) = \sum_{i=1}^{m} r_{iNn} \ln\left\{\Phi(\beta_i)\right\} + r_{iYn} \ln\left\{\Phi(-\beta_i)\right\} + r_{iNs} \ln\left\{\Phi(\beta_i - d')/\sigma_s\right\} + r_{iYs} \ln\left\{\Phi(d' - \beta_i)/\sigma_s\right\} \qquad (5)$$

where r_{ijk} is the number of j responses at criterion i to stimulus k, i = 1,2, . . . ,M; j = YES, NO; k = s (signal), n (noise). The parameter vector for this case is

$$x = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \cdot \\ \cdot \\ \beta_M \\ d' \\ \sigma_s \end{bmatrix} \qquad (6)$$

Note that we can easily obtain the parameter estimates under the constraints of equal-variance distributions by setting σ_s in the above expression to unity.

A function subroutine written to generate the parameter estimates is shown in Figure 5. It prompts the user for the number of different criterion levels, and a parameter indicating whether it is to be an equal-variance case or whether $B = 1/\sigma_s$ is a parameter to be estimated.

```
      FUNCTION CSTFN(X,NX,LFRST,MXITR,MXFGS,MXEXP,MXFLS,MXCTR,LPRT,DLM0)
C*****FUNCTION SUBROUTINE FOR MAXIMUM LIKELIHOOD ESTIMATION OF ROC PARAM
C*****WRITTEN FOR TELETYPE INTERACTION
      DIMENSION X(1),R(2,6),DLM0(1)
      GO TO (1,30),LFRST
C*****THIS PORTION FOR THE INITIALIZATION CALL TO INPUT DATA
    1 WRITE (1,1000)
1000  FORMAT (' READ NX,MXITR,MXFGS,MXEXP,MXFLS,MXCTR,LPRT')
      READ (1,1010) NX,MXITR,MXFGS,MXEXP,MXFLS,MXCTR,LPRT
1010  FORMAT (14)
      WRITE (1,1005)
1005  FORMAT (' NUMBER OF CRITERIA/1 FOR EQUAL VAR,2 OTHERWISE')
      READ (1,1010) MCRIT,LB
      IF ((LB-1)*(LB-2)) 1,5,1
    5 WRITE (1,1020)
1020  FORMAT (' READ #NO,#YES RESPONSES;FIRST NOISE, THEN S + N')
      DO 10 ICRIT = 1,MCRIT
      DO 10 ISIG = 1,2
      I = ISIG + 2*(ICRIT - 1)
   10 READ (1,1030) R(1,I),R(2,I)
1030  FORMAT (F10.2)
      WRITE (1,1040)
1040  FORMAT (' READ X0(I),DLM0(I)')
      IMX = MCRIT + LB
      DO 20 I = 1,IMX
   20 READ (1,1030) X(I),DLM0(I)
C*****NORMAL CALCULATIONS HERE
C*****STRIP D' FROM X AND LIMIT IF NECESSARY
   30 IX = MCRIT + 1
      IF (X(IX) - .01) 34,38,38
   34 X(IX) = .01
   38 D = X(IX)
C*****GET STD. DEV. RATIO
      GO TO (40,50),LB
   40 B = 1.
      GO TO 60
   50 IX = MCRIT + 2
C*****LIMIT ALLOWABLE B .GE. 0.01
      IF (X(IX) - .01) 54,58,58
   54 X(IX) = .01
   58 B = X(IX)
C*****CALCULATE LOG LIKELIHOOD FUNCTION (LESS CONSTANTS)
   60 XLLF = 0.
      DO 100 ICRIT = 1,MCRIT
      DO 100 ISIG = 1,2
      GO TO (70,80),ISIG
   70 Z = X(ICRIT)
      GO TO 90
   80 Z = B*(X(ICRIT) - D)
C*****THEORETICAL PROB('NO') FROM GAUSSIAN CUMULATIVE DISTRIBUTION
   90 P = GCDF(Z)
      IP = ISIG + 2*(ICRIT - 1)
  100 XLLF = XLLF + R(1,IP)*ALOG(P) + R(2,IP)*ALOG(1. - P)
C*****MINIMIZE THE NEGATIVE LOG LIKELIHOOD FUNCTION
      CSTFN = - XLLF
      RETURN
      END

      FUNCTION GCDF(Z)
      AZ = ABS(Z)
      T = 1./(1. + .2316419*AZ)
      D = .3989423*EXP( - Z*Z*.5)
      P = 1. - D*T*((((1.330274*T - 1.821256)*T + 1.781478)*T
     @ - .3565638)*T + .3193815)
      IF (Z) 1,2,2
    1 P = 1. - P
    2 GCDF = P
      RETURN
      END
```

Figure 5

For a test case, we assumed a value of x equal to ($\beta_1 = .5$, $\beta_2 = .75$, $\beta_3 = 1.25$, $d' = 1.00$, $\sigma_s = 1.25$). We used as responses the values proportional to the theoretical probabilities, i.e., we expect a "perfect" fit. Starting from the initial condition (−.5, 0, .5, .5, .5), the random leap algorithm converged to within .01 of the true value of each component of the parameter vector within 100 iterations. To determine how well these data from an unequal variance distribution are fit by an equal-variance model, we used the equal-variance option of the program and found the equal-variance fit to the unequal-variance data; the final estimates were ($\beta_1 = .531$, $\beta_2 = .756$, $\beta_3 = 1.1$, $d' = .96$), and the random leap algorithm, starting from the same initial conditions as before, again converged to within .01 of the final values after 100 iterations. Each of these solutions took about 3 min on the PDP-12 computer described above. Since a printout was made at each improvement, though, the computations were limited at times by the print operation.

Parameter Estimates from Grouped Data

The maximum likelihood procedure is one method of obtaining estimates of distribution parameters for grouped data, for example, obtaining the mean and standard deviation of a distribution from data such as is gathered in poststimulus histogram form. In these cases, the log likelihood function is proportional to

$$\ell(x) \propto \sum_{i=1}^{N} r_i \ln P_i(x) \tag{7}$$

where r_i is the number of responses in the ith group ($i = i,2, \ldots, N$), and P_i is the theoretical probability of the sample falling in the ith group. Under the assumption that the underlying distribution is Gaussian, P_i is given by

$$P_i = \Phi\{(c_{i+1} - \mu)/\sigma\} - \Phi\{(c_i - \mu)/\sigma\} \tag{8}$$

$$c_1 = -\infty, \; c_{N+1} = +\infty$$

and the parameter vector is

$$x = \begin{pmatrix} \mu \\ \sigma \end{pmatrix} \tag{9}$$

Simple changes can be made to the function routine shown in Figure 5 to perform these calculations. The boundaries of the groups $\{c_i\}$ must, of course, be read in during the initialization phase, but it is a simple matter to program Equation 7 and solve for the mean, μ, and standard deviations, σ.

An alternative to using the maximum likelihood estimation criterion is the minimum χ^2 criterion (Rao, 1973). This has many of the advantages of the maximum likelihood procedure, and serendipitously, one is calculating the χ^2 value which can be used in a goodness-of-fit test when the final estimates have been obtained.

Generalized Least Squares

One of the most common and most powerful forms of regression analysis is generalized least squares, which can be written in the form

$$f(x) = (\tilde{y} - h(x))^T W(\tilde{y} - h(x)) \tag{10}$$

where $\tilde{y}$ is a k-dimensional vector of observations (data), h is a vector function of the parameter x, and W is a positive semidefinite (symmetric) matrix of weighting coefficients. The objective is to choose the parameter vector x to minimize this cost function. In the linear form, Equation 10 becomes

$$f(x) = (\tilde{y} - Hx)^T W(\tilde{y} - Hx) \tag{11}$$

where H is a kxn matrix. This has a unique solution under suitable conditions on H and W (Rao, 1973), ususally satisfied in practice. One may use the random leap algorithm to find the solution to Equation 11 rather than go through the matrix inversions and manipulations required to solve for the vector x. We also note that if H is square and nonsingular, and if we solve Equation 11 with $\tilde{y}$ equal to zero except for a 1 in row i, then solution x is the ith column of the matrix H^{-1}. Thus, n separate parameter searches will completely invert the nxn matrix H.

CONCLUSIONS

The disadvantage of a small laboratory computer for postexperimental data processing is its small memory and inability to accommodate large sophisticated data processing programs, while the advantages of using the small computer are data compatibility, ease of program development, ability to run for long periods of time, low cost, and accessibility. Parameter optimization is one area which has received relatively little attention in small computer applications. Direct search methods are the easiest to program, and they require little core storage and no analytical gradient calculations.

Random search algorithms are one member of this class, and Rastrigan (1963) has shown that the average rate of convergence of the random creep algorithm may actually be superior to the gradient method in which the gradients are calculated numerically at each step. The slow rate of convergence experienced in the random creep algorithm is due to small step size and was overcome by the random leap algorithm proposed here. This algorithm operates by starting off in the global search mode and automatically reducing the step size as the search procedure approaches the extremum of the function. After reaching a minimum (or maximum), it

branches out and starts a global search by gradually expanding the size of the random increments from their original, moderately sized values.

The random leap algorithm was compared to the random creep method on a bimodal likelihood function and showed superior convergence characteristics. We then presented a function subroutine to be used with the random leap program to calculate the maximum likelihood estimates of receiver operating characteristic parameters in YES-NO tasks; this was followed by a description of maximum likelihood estimation of distribution parameters from grouped data (histograms). Generalized least squares regression was also considered, and it was shown how one could perform matrix inversion using successive applications of direct search methods.

REFERENCE NOTE

1. CHANDLER, J. P. *Subroutine STEPIT: An algorithm that finds the values of the parameters which minimize a given continuous function.* A copyrighted program. J. P. Chandler, Copyright, 1965.

REFERENCES

BEKEY, G. A., & UNG, M. T. A comparative evaluation of two global search algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 1974, **SMC-4**, 112-116.

DORFMAN, D. D., BEAVERS, L. L., & SASLOW, C. Estimation of signal detection theory parameters from rating method data: A comparison of the method of scoring and direct search. *Bulletin of the Psychonomic Society*, 1973, **1**, 207-208.

FAUREAU, R. R., & FRANKS, R. G. Statistical optimization. In: *Proceedings of the Second International Computer Conference*, 1958, 437-443.

FLETCHER, R., & POWELL, M. J. D. A rapidly convergent descent method for minimization. *The Computer Journal*, 1963, **6**, 163.

GREEN, D. M., & SWETS, J. A. *Signal detection theory and psychophysics*. New York: Wiley, 1966.

HOOKE, R., & JEEVES, T. A. Direct search solution of numerical and statistical problems. *Journal of the Association for Computer Machinery*, 1961, **8**, 212-229.

RAO, C. R. *Linear statistical inference and its applications* (2nd ed.). New York: Wiley, 1973.

RASTRIGAN, L. A. The convergence of random search method in the presence of noise. *Automation and Remote Control*, 1963, **24**, 1337-1342.

NOTE

1. This is easily done by intentionally overflowing the FORTRAN integer multiplication, and on a 12-bit machine, provides 1,024 uniformly distributed random numbers (-2047, 2047) before repeating the sequence.

(Received for publication November 27, 1974;
revision received April 1, 1975.)