An Automatic Method for finding the Greatest or Least Value of a Function

By H. H. Rosenbrock

The greatest or least value of a function of several variables is to be found when the variables are restricted to a given region. A method is developed for dealing with this problem and is compared with possible alternatives. The method can be used on a digital computer, and is incorporated in a program for Mercury.

1. Introduction

The problem of finding maxima or minima of a function arises in a number of ways. For example, one method[1] of solving the nonlinear simultaneous equations

$$f_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1, 2, \ldots, n \tag{1}$$

is to form the function

$$u(x) = \sum_{i=1}^{n} [f_i(x)]^2 \tag{2}$$

which will have a minimum (equal to zero) whenever equations (1) are satisfied. A second example[2] arises in finding the best operating conditions for a chemical process, where the function is determined experimentally.

The work described in this paper arose from a need to design chemical processes in such a way that they produce the most economical result. This usually will be defined to mean the cheapest product, taking into account the capital and operating costs of the plant, but may sometimes be defined in other ways. The problem is then to vary the available parameters (flows, pressures, temperatures, etc.) until the cost of the product is a minimum. The type of calculation which is needed in order to obtain the cost of product from the parameters is illustrated by an example in the Appendix. Here there are five parameters, and up to two or three times this number may be needed in similar problems.

A number of things are obvious from a brief study of the Appendix. The first is that it will be quite impracticable to differentiate the expressions in order to find the gradient at any point: this must be done numerically. Secondly, a method must be found for putting in limits* such as the inequality (63). These limit the possible solutions to a region in "parameter-space," and nearly always the best obtainable solution will lie on the boundary of this region. Thirdly, if there is strong "interaction" between the parameters[2] the problem will be a severe one, and it will be desirable to have a fully-automatic method of dealing with it on a digital computer.

In the next Section some of the difficulties will be discussed. The development of a practical solution is described in Section 3.

* "Limit" is used throughout in its ordinary, and not in its special mathematical sense.

2. Preliminary Discussion

The problem discussed in the last Section can be stated in the following way. A function $u(x)$ of the parameters $x_1, x_2, \ldots, x_n$ is to be a maximum or minimum (for definiteness we take the latter) subject to

$$\left.\begin{array}{ll} g_1(x_1, x_2, \ldots, x_n) & \leqslant x_1 \leqslant h_1(x_1, x_2, \ldots, x_n) \\ g_2(x_1, x_2, \ldots, x_n) & \leqslant x_2 \leqslant h_2(x_1, x_2, \ldots, x_n) \\ \ldots\ldots & \\ g_n(x_1, x_2, \ldots, x_n) & \leqslant x_n \leqslant h_n(x_1, x_2, \ldots, x_n) \end{array}\right\} \tag{3}$$

$$\left.\begin{array}{l} g_{n+1}(x_1, x_2, \ldots, x_n) \leqslant x_{n+1}(x_1, x_2, \ldots, x_n) \\ \qquad \leqslant h_{n+1}(x_1, x_2, \ldots, x_n) \\ g_{n+2}(x_1, x_2, \ldots, x_n) \leqslant x_{n+2}(x_1, x_2, \ldots, x_n) \\ \qquad \leqslant h_{n+2}(x_1, x_2, \ldots, x_n) \\ \ldots\ldots \\ g_l(x_1, x_2, \ldots, x_n) \leqslant x_l(x_1, x_2, \ldots, x_n) \\ \qquad \leqslant h_l(x_1, x_2, \ldots, x_n) \end{array}\right\} \tag{4}$$

In any given problem many of the inequalities (3) may be absent in the formulation, and there will usually be a small number (say three or four) of inequalities (4). The reason for keeping the first set separate and complete, instead of including them in the second (more general) form is to exclude meaningless solutions. Even though they may not be stated explicitly in the first formulation of the problem, there are almost always engineering restrictions on the parameters. These may become important in defining the solution, and it is therefore desirable to include them in the specification. We thus avoid, for example, gas compositions with negative amounts of one constituent, or pressures outside the range of existing technology. Where a limit would be meaningless, it can always be made a very large or very small constant, or can be stated in the form, for example,

$$g_1 = x_1 - 1 \tag{5}$$

Leaving aside for the present the limits (3) and (4), the simplest way of finding a minimum of $u(x)$ would be to change $x_1, x_2, \ldots, x_n$ in turn, reducing u as far as possible with each variable and then passing on to the

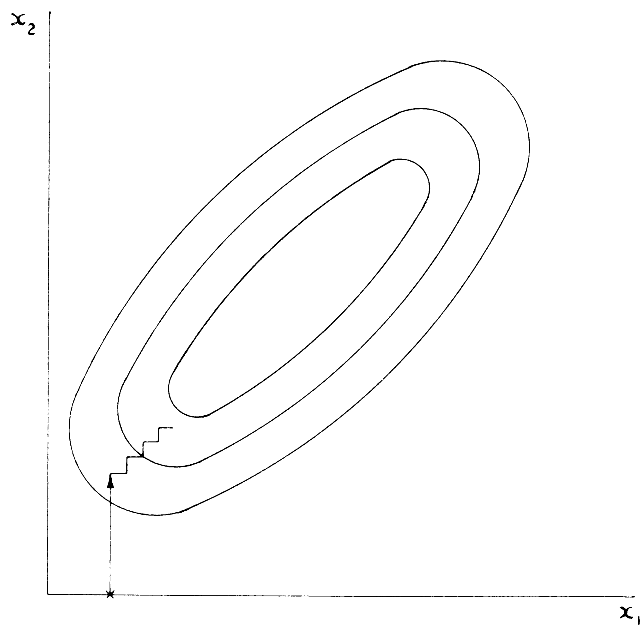

Fig. 1.—Method of changing one parameter at a time

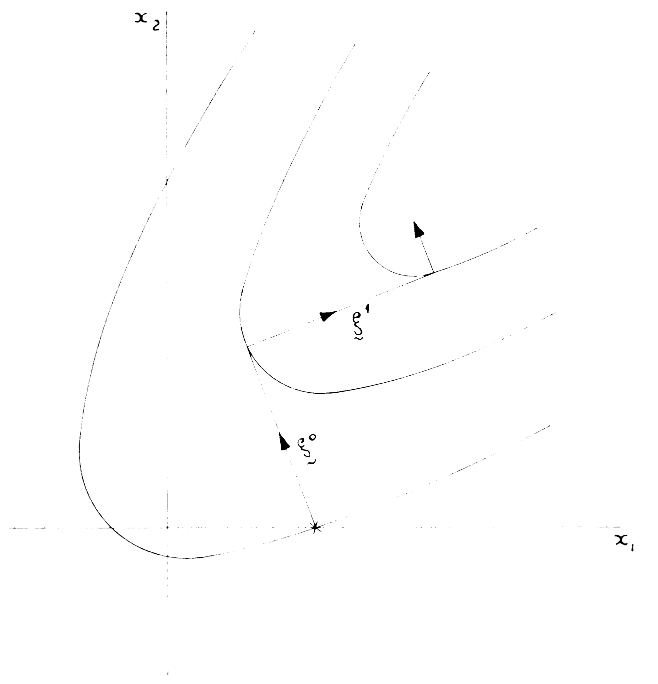

Fig. 2.—Method of steepest descent

next. This is illustrated for two parameters in Fig. 1. The method works well if the contours of constant u are nearly circular, but for an obvious reason becomes very slow when there is interaction. In two dimensions, the existence of interaction corresponds to the presence of a long, narrow ridge. If this is not parallel to one of the two axes, progress can only be made by small steps in x_1 and x_2 in turn.

Because of this well-known difficulty, the method of steepest descent is usually recommended. In its pure form, this consists in finding the direction of steepest descent from

$$\xi_i = -\frac{\partial u}{\partial x_i}\Big/\left\{\sum_{i=1}^{n}\left(\frac{\partial u}{\partial x_i}\right)^2\right\}^{\frac{1}{2}} \tag{6}$$

where ξ_i are the components of a unit vector ξ^0 in the required direction. The value of u is then calculated at new points along a line from the first point parallel to ξ^0, until the least value is attained. Starting from this lowest point, the process is repeated: equation (6) is evaluated again and progress is made along a line parallel to the new vector ξ^1.

At first sight this process seems attractive, but the apparent advantages are easily seen to be in large part illusory. The vector ξ^0 will be perpendicular to the contour at the starting-point (Fig. 2), and progress will be made until the local contour is parallel to ξ^0. At this point ξ^1 will be found, and it is perpendicular to the local contour, and therefore to ξ^0. Similarly ξ^2 will be perpendicular to ξ^1 and hence parallel to ξ^0. Thus with two variables the method of steepest descent is equivalent to the method of changing one parameter at a time. The two directions which will be used are fixed once for all by the choice of starting point, and need bear no relation to the direction of any ridges that may be present. If the contours are nearly circular, the method of steepest descent will have a small advantage, but this has no practical importance.

When there are more than two variables, the two methods are no longer equivalent: each vector ξ in the method of steepest descent is normal to the preceding vector, but n successive vectors do not necessarily form a mutually orthogonal set. Nevertheless, there is no reason to believe that the method of steepest descent will have much advantage: certainly it will not if the difficulty in a problem arises chiefly from the interaction of only two variables.

To overcome this defect, it has been proposed[1] that the method of steepest descent should be modified. The vector ξ^0 is found as before, and progress is made parallel to it until the least value is attained. The distance from the starting point is then multiplied by 0·9, and ξ^1 is evaluated at the point so found (Fig. 3). After, say, four repetitions of this procedure one step of full length is taken.

This would be a possible method for our problem, but there is a further difficulty. The partial derivatives of u must be obtained numerically at the beginning of each step. This can be done by evaluating u at two neighbouring points and using, for example,

$$\frac{\partial u}{\partial x_1} \doteqdot \frac{u(x_1 + h, x_2, \ldots, x_n) - u(x_1, x_2, \ldots, x_n)}{h} \tag{7}$$

Here h must be small enough so that $\frac{h^2}{2}\frac{\partial^2 u}{\partial x_1^2}$ can be neglected. It will be found later that a simple way to deal

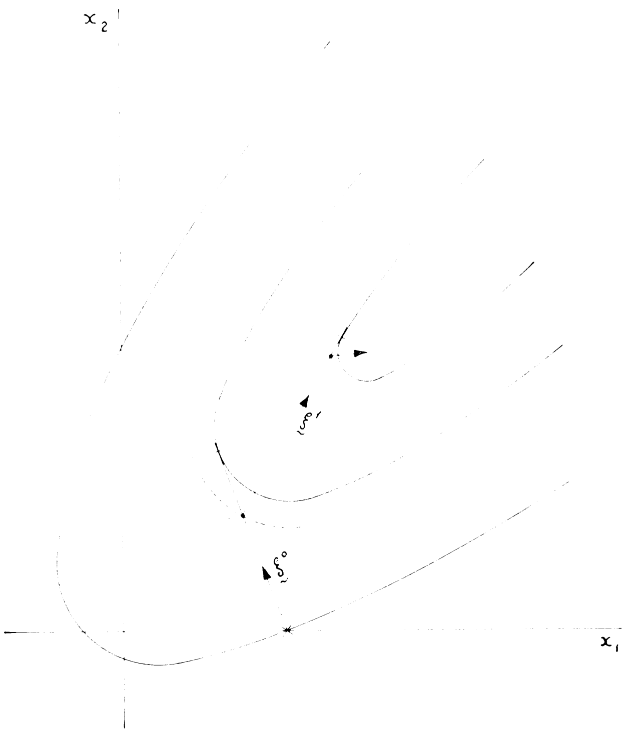

Fig. 3.—Modified method of steepest descent

with the limits (3) and (4) is to modify u near the boundaries of the permitted region. When this is done the last restriction limits h to, say, the last two significant figures if we are working with eight decimal digits. If h is as small as this, however, we shall fail to detect any value of $\partial u/\partial x$ which is less than about 1/200.

Because of these difficulties with the method of steepest descent a new method was developed. It is better suited than the former to automatic calculation on a digital computer. It also turns out to be faster in some circumstances.

3. Development of New Method

There are three difficulties which have to be met in developing a practical method for dealing with the proposed problem. These are dealt with separately below.

3.1 *Determining Length of Step*

The simplest problem is to decide the length of step to be taken in the desired direction, assuming this direction to be known. The principle adopted was to try a step of arbitrary length e. If this succeeded, e was multiplied by $\alpha > 1$. If it failed, e was multiplied by $-\beta$ where $0 < \beta < 1$. "Success" here was defined to mean that the new value of u was less than or equal to the old value. Thus if e was initially so small that it made no change in u, it would be increased on the next attempt. Each such attempt will be called a "trial."

3.2 *Determining Direction of Step*

The next problem is to decide when and how to change the directions ξ in which the steps e are taken. It was decided to work throughout with n orthogonal directions $\xi_1, \xi_2, \ldots, \xi_n$, rather than choose a single direction in which to progress at each stage. It is necessary anyway to examine neighbouring points in each of n directions, in order to determine the best direction of advance. If one of the points examined in this way makes u less than the previous value, we might as well accept it as a new starting point.

It was also decided to make one trial, of the kind described in Section 3.1, in each of the n directions in turn. An alternative procedure would be to make the number of trials in any direction depend on their result. It was thought that this would not generally speed up the process, but no systematic test was made of this assumption.

A number of different methods were tried for determining the point at which to compute new directions ξ. The one finally chosen was to go on until at least one trial had been successful in each direction, and one had failed. It will be noticed that a trial must in the end succeed because e becomes so small after repeated failures that it causes no change in u. The set of trials made with one set of directions, and the subsequent change of these directions, will be called a "stage."

The method chosen for finding the new directions of ξ was the following. Suppose that d_1 is the algebraic sum of all the successful steps e_1, in the direction ξ_1, etc. Then let

$$\left.\begin{aligned} \boldsymbol{A}_1 &= d_1\xi_1^0 + d_2\xi_2^0 + \ldots + d_n\xi_n^0 \\ \boldsymbol{A}_2 &= \qquad\quad\; d_2\xi_2^0 + \ldots + d_n\xi_n^0 \\ &\cdot\;\cdot\;\cdot\;\cdot\;\cdot \\ \boldsymbol{A}_n &= \qquad\qquad\qquad\qquad\;\; d_n\xi_n^0 \end{aligned}\right\} \tag{8}$$

Thus $\boldsymbol{A}_1$ is the vector joining the initial and final points obtained by use of the vectors $\xi_1^0, \xi_2^0, \ldots, \xi_n^0$, $\boldsymbol{A}_2$ is the sum of all the advances made in directions other than the first, etc.

Orthogonal unit vectors $\xi_1^1, \xi_2^1, \ldots, \xi_n^1$, are now obtained in the following way:

$$\left.\begin{aligned} \boldsymbol{B}_1 &= \boldsymbol{A}_1 \\ \xi_1^1 &= \boldsymbol{B}_1/|\boldsymbol{B}_1| \\ \boldsymbol{B}_2 &= \boldsymbol{A}_2 - \boldsymbol{A}_1 \,.\, \xi_1^1\xi_1^1 \\ \xi_2^1 &= \boldsymbol{B}_2/|\boldsymbol{B}_2| \\ &\cdot\;\cdot\;\cdot\;\cdot\;\cdot \\ \boldsymbol{B}_n &= \boldsymbol{A}_n - \sum_{j=1}^{n-1} \boldsymbol{A}_n \,.\, \xi_j^1\xi_j^1 \\ \xi_n^1 &= \boldsymbol{B}_n/|\boldsymbol{B}_n| \end{aligned}\right\} \tag{9}$$

No ambiguity is likely to arise, since the method used ensures that no d can be zero. It is of course possible that one or more of the d are so small that they are lost in the summations of equations (8), but this is unlikely in practice. The result of applying equations (8) and (9) several times is to ensure that ξ_1 lies along the direction

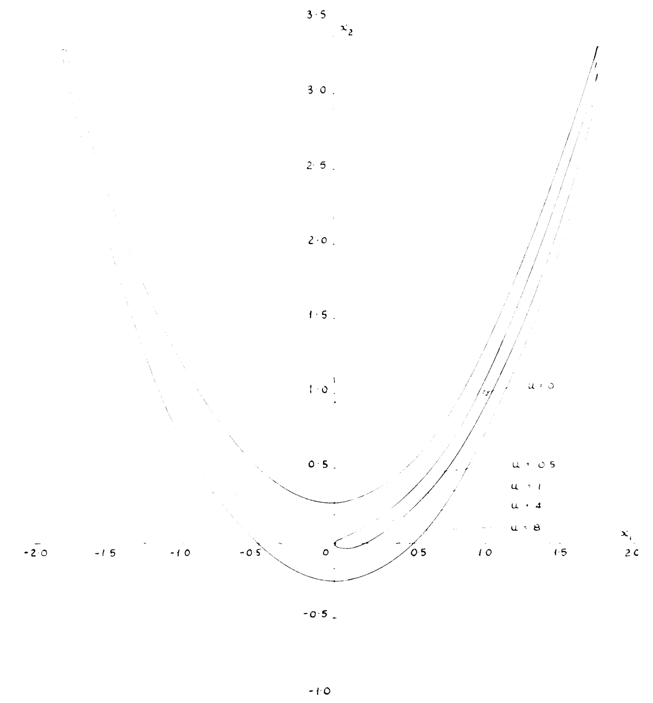

Fig. 4.—Contours of the function used in developing the program

of fastest advance, ξ_2 along the best direction which can be found normal to ξ_1, and so on.

The numerical work of developing this process was carried out with the function

$$u(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{10}$$

This has a minimum value $u = 0$ at $x_1 = x_2 = 1$, with a curved valley following the parabola $x_2 = x_1^2$ as shown in Fig. 4. Computer runs were started from $x_1 = -1·2$, $x_2 = 1$, so that the current point had to descend into the valley, and then follow it around its curve to the point (1, 1). Two hundred trials were used, and the values of x_1, x_2 and u were printed out after each stage. The initial values of e_1 and e_2 were $+0·1$, with ξ_1 parallel to the axis of x_1, and ξ_2 parallel to the axis of x_2.

A set of calculations was made to determine the best values of α and β. Table 1 shows the values of x_1, x_2 and u achieved after 200 trials with given values of α and β. Too great refinement is not justified, as the results must depend to some extent on the particular problem. The values $\alpha = 3$, $\beta = 0·5$ were therefore adopted.

For comparison, 200 trials were made without changing the directions ξ: this is the first process mentioned in Section 2. The result was

$$x_1 = -0·970, \quad x_2 = 0·945, \quad u = 3·882.$$

The method of steepest descent was tried first in its simple form. The direction was obtained analytically from equation (6), and the length of step was determined by the method of Section 3.1, with $\alpha = 3$ and $\beta = 0·5$. Trials in any given direction were continued until there had been at least one success, followed by five failures. This ensured that the least value along the chosen line was found with reasonable accuracy. After 200 trials the result was

$$x_1 = -0·605, \quad x_2 = 0·371, \quad u = 2·578.$$

Table 1

α	β	x_1	x_2	u
1·5	0·2	−0·004	−0·015	1·030
1·5	0·5	0·416	0·212	0·290
1·5	0·7	0·989	0·977	0·00027
1·5	0·8	0·964	0·929	0·00135
2	0·3	0·448	0·201	0·305
2	0·5	0·987	0·972	0·00043
2	0·7	0·983	0·963	0·00107
2	0·8	0·970	0·946	0·00280
3	0·2	0·752	0·557	0·068
3	0·4	0·980	0·962	0·00050
3	0·5	0·995	0·991	0·000022
3	0·7	0·928	0·854	0·0113
3	0·8	0·843	0·712	0·025
5	0·2	0·717	0·523	0·089
5	0·3	0·962	0·927	0·00186
5	0·4	0·885	0·776	0·0160
5	0·5	0·99989	0·99978	0·00000001

In comparing this last result with those quoted in Table 1, some uncertainty arises because of the method used for determining the point at which to compute a fresh vector ξ—some other method might have produced a better result in 200 trials. This would certainly be true if one of the available methods[1] for interpolating or extrapolating was used to find the lowest point along the chosen line of advance. We exclude this possibility because it would lead to great difficulty with the method used later for putting in limits. The best way of comparing results then seems to be on the basis of an equal number of stages for each method.

The results in Table 1 for $\alpha = 3$, $\beta = 0·5$ were obtained in 21 stages (one incomplete). A further test of the method of steepest descent was therefore made in which 21 stages were used, and each stage was terminated when at least one success had been followed by ten failures. This ensured that the least value along the chosen direction was found with good accuracy. The result was

$$x_1 = -0·235, \quad x_2 = 0·068, \quad u = 1·542$$

and 340 trials were needed.

Another test was made with the method of steepest descent, in which the progress in each stage was reduced to nine-tenths of its original value before a new vector ξ was calculated. Ten failures were again allowed in every stage. In 21 stages (338 trials) the result obtained was

$$x_1 = 0·219, \quad x_2 = 0·046, \quad u = 0·611.$$

The new process with $\alpha = 3$, $\beta = 0 \cdot 5$ made slightly more progress than this (to $u = 0 \cdot 569$) in nine stages. Finally, a test was made in which four successive stages were reduced by the factor $0 \cdot 9$, and the next was of full length. Allowing ten failures in each stage the result after 21 stages (344 trials) was

$$x_1 = -0 \cdot 180, \quad x_2 = 0 \cdot 036, \quad u = 1 \cdot 393.$$

A further comparison was made using a problem which has been studied by Booth.[3] This is solve the equations

$$\left.\begin{aligned} x_1 + 2x_2 &= 7 \\ 2x_1 + x_2 &= 5 \end{aligned}\right\} \qquad (11)$$

by finding the minimum of

$$u = (x_1 - 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \qquad (12)$$

starting from $x_1 = x_2 = 0$. The required answer is $x_1 = 1$, $x_2 = 3$, and Booth quotes the following results for three different methods:

(i) "Southwell's method" in 18 stages gave

$$x_1 = 0 \cdot 98, \quad x_2 = 3 \cdot 02, \quad u = 0 \cdot 0008.$$

(ii) The "tangent descent" method in 5 stages gave

$$x_1 = 1 \cdot 04, \quad x_2 = 2 \cdot 97, \quad u = 0 \cdot 003.$$

(iii) The "interpolative descent" method in 4 stages gave

$$x_1 = 1 \cdot 00, \quad x_2 = 3 \cdot 00, \quad u = 0.$$

In four stages (44 trials) the new method gave

$$x_1 = 1 \cdot 040, \quad x_2 = 2 \cdot 970, \quad u = 0 \cdot 0028.$$

After five stages (54 trials) the result was

$$x_1 = 1 \cdot 017, \quad x_2 = 2 \cdot 991, \quad u = 0 \cdot 00065.$$

Six stages (59 trials) gave

$$x_1 = 0 \cdot 996, \quad x_2 = 3 \cdot 0009, \quad u = 0 \cdot 00001.$$

For the last comparison, the introductory section of a problem quoted by Box and Coutie[4] was used. This involves the adjustment of parameters in two simultaneous differential equations until the computed solution agrees as well as possible with experimental results. The Runge–Kutta routine available with the Mercury autocode was used for solving the differential equations: 320 time-steps were used for each solution. The variables x_1 and x_2 were θ_1 and θ_2 of Ref. 4, and their starting values were $x_1 = 1 \cdot 19$, $x_2 = 1 \cdot 19$, with $e_1 = e_2 = +0 \cdot 1$. Using the method of steepest descent, Box and Coutie, after integrating the differential equations 23 times, find a minimum at

$$x_1 = 1 \cdot 0779, \quad x_2 = 0 \cdot 8061, \quad u = 0 \cdot 00738.$$

They interpolate at the end of each of the three stages (which gives some advantage in the comparison), but the interpolation in the last stage has been neglected here. The new method, after 24 integrations (three stages, one incomplete), gave

$$x_1 = 1 \cdot 0759, \quad x_2 = 0 \cdot 8196, \quad u = 0 \cdot 00739$$

and 17 further trials failed to reduce this value of u. The time taken on the computer was 8 minutes for 41 trials, including the printing of x_1, x_2 and u at each trial.

We may conclude from these examples that the proposed method is not significantly slower than the available alternatives in simple problems (the possibility of interpolating or extrapolating not being admitted). In difficult problems it may be a good deal faster. It is well adapted to automatic calculation, and is not easily upset by minor irregularities in u (for example, by asymmetrical ridges). It does rely, as do the alternative methods, upon the continuity and smoothness of u.

3.3 *Inserting Limits*

It is slightly easier in dealing with limits to think about maxima rather than minima, and most of the development work with limits was done with the following simple problem:

A rectangular parcel is to be sent by post. The length must not exceed 3 ft. 6 in., and the length and girth combined must not exceed 6 ft. What shape of parcel gives the greatest volume? (24 in. × 12 in. × 12 in. = 3,456 cu. in.)

This gives

$$\left.\begin{aligned} &u = x_1 x_2 x_3 \\ &0 \leqslant x_1 \leqslant 42 \\ &0 \leqslant x_2 \leqslant 42 \\ &0 \leqslant x_3 \leqslant 42 \\ &0 \leqslant x_1 + 2x_2 + 2x_3 \leqslant 72. \end{aligned}\right\} \qquad (13)$$

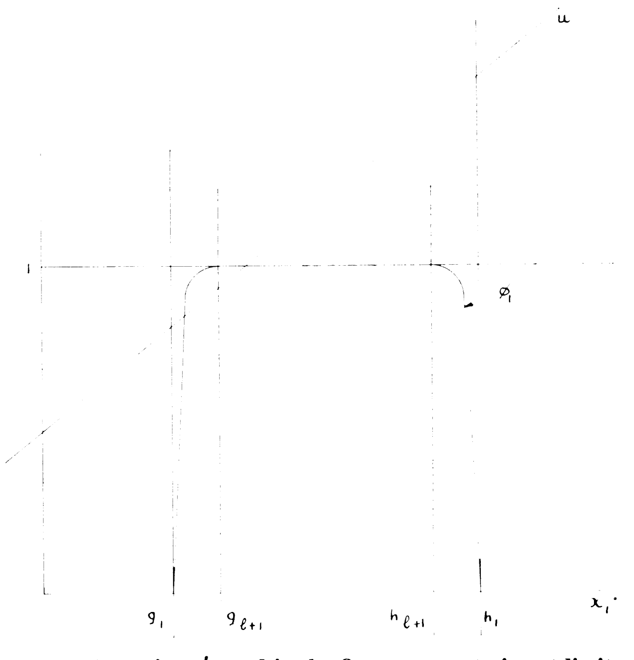

Fig. 5.—Function ϕ used in the first attempt to insert limits

where x_1, x_2 and x_3 are respectively the length, breadth and depth, and u is to be as large as possible. The inequalities have been completed in an obvious way.

One way to deal with the inequalities is to multiply u (which is positive) by l functions, each of which has the following properties:

(i) It is equal to 0 if its associated variable is outside the limits, for example if $x_1 < 0$ or $x_1 > 42$.

(ii) It is equal to 1 when the variable is within the permitted range, from which two narrow boundary regions are excluded: for example when $0 \cdot 0001 < x_1/42 < 0 \cdot 9999$.

(iii) Within the narrow boundary regions it goes parabolically from 1 to 0.

This is illustrated in Fig. 5. If the boundary regions for x_i extend from g_i to g_{l+i} and from h_{l+i} to h_i, the function described is

$$\left.\begin{aligned} \phi_i &= 0,\ x_i \leqslant g_i \\ \phi_i &= 1 - \left(\frac{g_{l+i} - x_i}{g_{l+i} - g_i}\right)^2,\ g_i < x_i < g_{l+i} \\ \phi_i &= 1,\ g_{l+i} \leqslant x_i \leqslant h_{l+i} \\ \phi_i &= 1 - \left(\frac{x_i - h_{l+i}}{h_i - h_{l+i}}\right),\ h_{l+i} < x_i < h_i \\ \phi_i &= 0,\ h_i \leqslant x_i. \end{aligned}\right\} \quad (14)$$

The product

$$u' = \phi_1 \phi_2 \ldots \phi_l u \quad (15)$$

will then be zero if any variable does not satisfy the limits, and will equal u if all the variables are in their permitted range and not in the boundary zone. Within the boundary zone u' goes from u to 0. The maximum value of u' will be a good approximation to the required answer if the boundary zone is narrow enough.

To see how narrow the boundary zone should be, we consider a function u of one variable x_1. Near a maximum within the interior region $g_{l+1} \leqslant x_1 \leqslant h_{l+1}$,

$$u = u_m - A(x_1 - x_{1m})^2 \quad (16)$$

where u_m is the maximum value of u and x_{1m} is the corresponding value of x_1. If the Mercury autocode[5] is used, all quantities will be held in floating-point form with an accuracy of about 8 decimal digits. If u_m and x_{1m} are scaled to unit magnitude (which does not affect the problem), and if u is less than u_m by one unit in the last decimal place, then

$$\begin{aligned} A(x_1 - x_{1m})^2 &\doteqdot 10^{-8} \\ x_1 - x_{1m} &\doteqdot 10^{-4} A^{-1/2} \end{aligned} \quad (17)$$

Depending on the value of A, we can therefore find x_{1m} with the following accuracy:

about two decimal places if	$A = 10^{-4}$
three	$A = 10^{-2}$
four	$A = 1$
five	$A = 10^2$, etc.

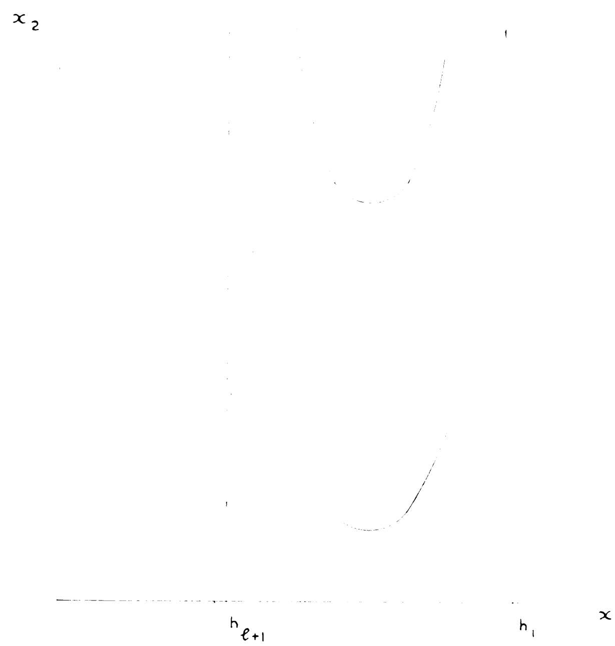

Fig. 6.—Long narrow ridge produced by method of inserting limits

In practice we may expect to get three decimal places or more on most occasions, but sometimes only two decimal places. The extension when there are several variables, $x_1, x_2, \ldots, x_n$, is obvious.

If now the maximum is in the boundary zone, and if this has a width of about $10^{-4}x_{1m}$, there will be no significant loss of accuracy in the final value of x_1 from the use of the zone. The value of u will be in error by an amount depending on its gradient at the boundary, but we can expect to get an accuracy of three decimal figures in u on many occasions. Since the values of x_{1m}, x_{2m}, etc., are not at first known, our criterion would lead us to use a width of boundary zone equal to $10^{-4}|h_i|$ or $10^{-4}|g_i|$, whichever was the greater. In many cases, however, it will be good enough to use $10^{-4}(h_i - g_i)$ and this was done in the following work.

The problem already described was now attacked, using

$$g_{l+i} - g_i = h_{l+i} - h_i = 10^{-4}(h_i - g_i). \quad (18)$$

A difficulty was immediately found, and it is illustrated in Fig. 6. This shows the boundary region of a function $u(x_1, x_2)$ which has been treated in the above way. There is a long, narrow, slowly rising ridge, and the contours are drawn at intervals of one unit in the last decimal place of u'. Between two contours the machine finds successive values of u' which are equal, and these trials count as successes. The current point will wander about within a contour, more or less at random, until it chances to find a higher value of u'. Until this happens there is nothing to show what direction should be preferred. Evidently, if the ridge rises slowly enough, the probability of further progress will become very small.

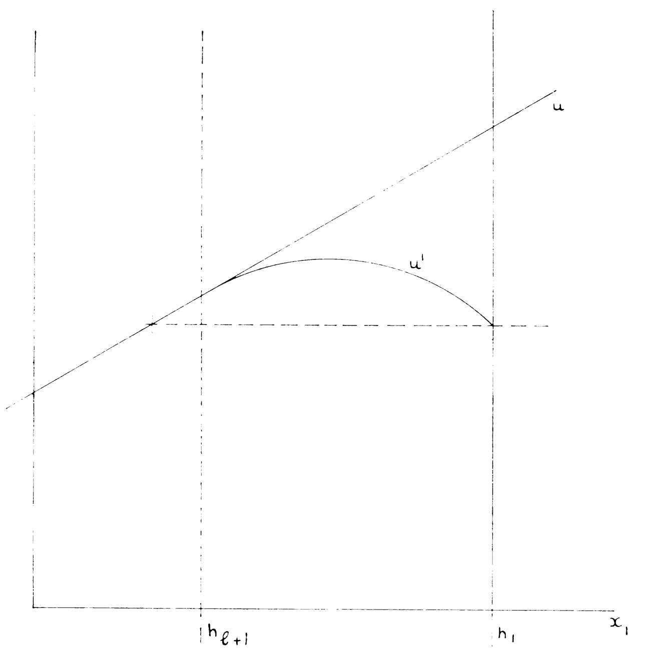

Fig. 7.—Modified procedure for inserting limits

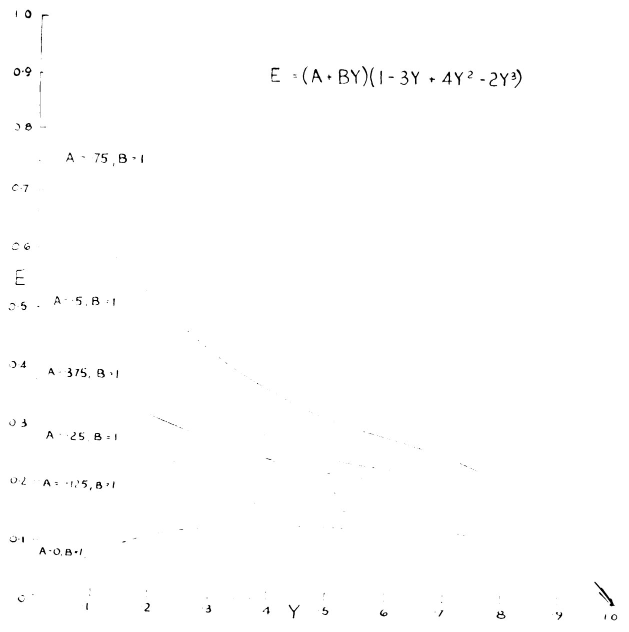

Fig. 8.—The function E in the boundary zone

This will always happen as the maximum of u' is approached, and progress may stop at a considerable distance from the required point. In the problem described, 600 trials (that is, 200 for each variable) gave

$$x_1 = 23\cdot 794, \quad x_2 = 12\cdot 257, \quad x_3 = 11\cdot 842,$$
$$u = 3{,}453\cdot 75431$$

and u had not increased after the 337th trial.

A number of methods were tried in an attempt to remove this difficulty. For example, the width of the boundary zone was made large at first and decreased as the solution proceeded. This was unsuccessful, apparently because the line to be followed moved sideways, making it difficult for the current point to advance.

A more promising method was to reduce u' at the boundary not to zero but to the highest value of u so far found at a permitted point not in the boundary zone. This is illustrated in Fig. 7. In practice the method works less well than would be expected, as the current point often enters the boundary zone at an early stage and then remains inside it.

The method finally adopted was to force the current point out of the boundary region of each x_i after a certain amount of progress had been made. When the current point left the boundary region a parameter g_{2l+i} or h_{2l+i} was set to the highest value u_0 so far found for u. Thus the following process was carried out for each x_i on every trial:

(i) Count the trial as "failed" if $x_i \leqslant g_i$ or $h_i \leqslant x_i$.
(ii) Leave u unchanged if $g_{l+i} \leqslant x_i \leqslant h_{l+i}$ and put $g_{2l+i} = h_{2l+i} = u_0$.
(iii) If $g_i < x_i < g_{l+i}$, replace u by

$$u' = g_{2l+i} + (u - g_{2l+i})(1 - 3\gamma + 4\gamma^2 - 2\gamma^3) \quad (19)$$

where
$$\gamma = \frac{g_{l+i} - x_i}{g_{l+i} - g_i}. \quad (20)$$

(iv) If $h_{l+i} < x_i < h_i$, replace u by

$$u' = h_{2l+i} + (u - h_{2l+i})(1 - 3\eta + 4\eta^2 - 2\eta^3) \quad (21)$$

where
$$\eta = \frac{x_i - h_{l+i}}{h_i - h_{l+i}}. \quad (22)$$

Within the boundary region we can write, approximately,

$$u - g_{2l+i} = A + B\gamma \quad (23)$$

or
$$u - h_{2l+i} = A + B\eta \quad (24)$$

as the case may be, where A and B are constants. Then for example,

$$E = (A + B\gamma)(1 - 3\gamma + 4\gamma^2 - 2\gamma^3) \quad (25)$$

has the following properties:

(i) When $\gamma = 0$, $E = A + B\gamma$, so that $u' = u$.
(ii) When $\gamma = 1$, $E = 0$, and $u' = g_{2l+i}$. Since the successive values of u_0 form a non-decreasing sequence, and g_{2l+i} is equal to a previous value of u_0, u' is less than or equal to the highest value of u so far found.
(iii) If $A = 0$,

$$E = \tfrac{1}{8}A[1 - (1 - 2\gamma)^4] \quad (26)$$

so that at $\gamma = \frac{1}{2}$, $E' = E'' = E''' = 0$.
(iv) For a given B, as A increases, the maximum of E moves towards smaller values of γ, as shown in Fig. 8.

Suppose now that the current point is within the

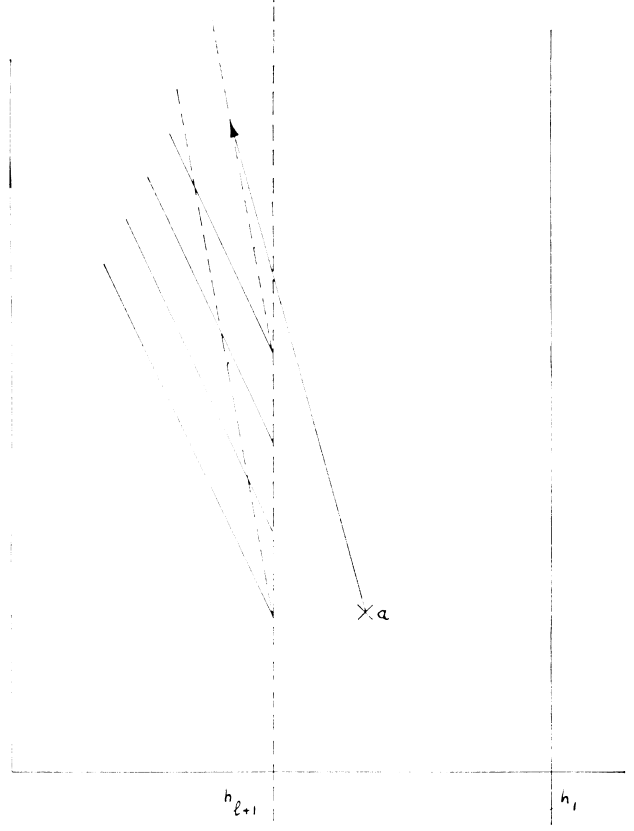

Fig. 9.—Behaviour of current point on emerging from the boundary zone

boundary region for x_1, at the point a in Fig. 9. It finds a sloping ridge which, by equation (26), will have a flat top. This makes it easy for progress to be made along the ridge, and the vector ξ_1 will after a time be lined up with the ridge. The current point will therefore issue from the boundary region into the interior part, and two things can then happen:

(i) If the ridge slopes upward steeply enough, as shown by the full-line contours in Fig. 9, progress will be possible in the direction of ξ_1.

(ii) If the ridge slopes only gradually, as shown by the broken lines in Fig. 9, progress in the direction of ξ_1 will not be possible in the interior of the region. The value of g_{2l+1} or h_{2l+1} will, however, have been re-set when the point leaves the boundary region. It can therefore re-enter this region, and will do so with a small value of A. The process can thus continue.

It is not necessary for u to be positive, and the procedure described above can therefore be used for minima simply by changing the sign of u. This is done internally by the program described later.

This method was applied to the example, with $g_{l+i} - g_i = 0\cdot0001(h_i - g_i)$, etc., and the result after 600 trials was

$$x_1 = 23\cdot9873, \quad x_2 = 12\cdot0026, \quad x_3 = 12\cdot0017, \quad u = 3{,}455\cdot09.$$

When the same problem was solved with the starting points 15, 10, 10 and 5, 10, 10 instead of 10, 10, 10, the results were respectively,

$$x_1 = 24\cdot0086, \quad x_2 = 11\cdot9990, \quad x_3 = 11\cdot9936, \quad u = 3{,}455\cdot00,$$
$$x_1 = 23\cdot9984, \quad x_2 = 11\cdot9996, \quad x_3 = 11\cdot9987, \quad u = 3{,}455\cdot05.$$

In this example only one limit, namely $x_1 + 2x_2 + 2x_3 \leqslant 72$ is effective. As a second example x_1 was limited to 20 in. and x_2 to 11 in. Here three limits are effective, and the result after 600 trials was

$$x_1 = 19\cdot9987, \quad x_2 = 10\cdot9989, \quad x_3 = 14\cdot9979, \quad u = 3{,}298\cdot83.$$

The correct result is

$$x_1 = 20, \quad x_2 = 11, \quad x_3 = 15, \quad u = 3{,}300.$$

A solution was found for the problem given in the Appendix using a late (but not the final) form of the program. It is difficult to be quite certain that the result found could not be improved, but from the engineering point of view it was satisfactory. A total of 1,000 trials was used.

4. Computer Program

The final program is available in Mercury autocode. Chapters 1 and 2 contain the organization for seeking the best result. A third chapter is added to define the particular problem, and chapter 0 contains data.

Chapter 3 must contain instructions for calculating u_n from $x_1, x_2, \ldots, x_n$. It must also compute any of the g_i or h_i which are not constant, and all of x_{n+1}, $x_{n+2}, \ldots, x_l$. Chapter 0 contains starting values for $x_1, x_2, \ldots, x_n$, and for $g_1, g_2, \ldots, g_l$ and $h_1, h_2, \ldots, h_l$. The values of n and l, and the number of trials k for each variable must be given. An index is set to 1 for a maximum and to -1 for a minimum, and the number of decimal places in the print-out of x_i and of u must be specified.

After each stage results are printed in the form

$$\begin{array}{llll} d_0 & u_0 & b_1 & b_2 \\ x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{array}$$

Here d_0 is the number of trials, u_0 is the best value so far found for u', and $x_1, x_2, \ldots, x_n$ are the corresponding x_i. The progress during the stage is b_1:

$$b_1 = |A_1|$$

and b_2 is defined by

$$b_2 = |A_2| \div |A_1|$$

where A_1 and A_2 are given by equations (8). If progress is steadily in one direction, b_2 will be small, and if b_2 remains generally less than 0·3 it should be suspected that the best value has not yet been reached, even though b_1 may be small.

The time taken will be about T sec, where

$$T = kn(0{\cdot}6 + t)$$

and t is the time in seconds to compute chapter 3 once.

5. Conclusions

The program described in Section 4 will find the greatest or least value of a function in an arbitrarily restricted region, and it may be useful as it stands for some practical applications. It is most unlikely that the process given here cannot be improved, and in work directed to this end the program may be useful as a standard of comparison.

The work showed forcibly the virtues of autocode. Well over fifty different programs were run in the course of development, and this would have been impracticable in any other way.

6. Acknowledgement

Acknowledgement is made to the Directors of Constructors John Brown Ltd. for permission to publish this paper.

7. Appendix—Example

Given x_f, E, H_c, H_h, P, S_c, S_h, K_c, β_c and β_h, find the least value of $u(R, e_R, e_h, G, L_1)$, where

$$u = \frac{0{\cdot}0216\sqrt{\dfrac{G}{EPx_f}}(N_R + N_C + N_h)}{R} + \frac{0{\cdot}051G}{EPx_f} \quad (30)$$

and

$$x_w = (1 - R)x_f \quad (31)$$

$$x_p = Ex_f \quad (32)$$

$$L = \frac{P(x_p - x_f)}{x_f - x_w} \quad (33)$$

$$Y_R = \frac{e_R x_w}{\beta_h} \quad (34)$$

$$m_R = \frac{L(1 + S_h)}{G(1 + H_h)} \quad (35)$$

$$Y_0 = Y_R + m_R(x_f - x_w) \quad (36)$$

$$x_h = \frac{\beta_h Y_0}{e_h} \quad (37)$$

$$L_4 = (L_1 - P)(1 - H_h S_c) + G(H_h - H_c) \quad (38)$$

$$L_0 = L_1 + G(H_h - H_c) \quad (39)$$

$$x_c = \frac{Px_f + L_4 x_h}{L_0} \quad (40)$$

$$Y_c = \frac{Y_R(1 + H_h)}{1 + H_c} - \frac{x_c(H_h - H_c)}{1 + H_c} \quad (41)$$

$$X_c = \frac{x_c(1 + S_h)}{1 + S_c} + \frac{Y_c(S_c - S_h)}{1 + S_c} \quad (42)$$

$$b_R = Y_R - m_R m_w \quad (43)$$

$$N_R = \left[\log\frac{(1 - \beta_h m_R)x_w - \beta_h b_R}{(1 - \beta_h m_R)x_f - \beta_h b_R}\right]\frac{1}{\log \beta_h m_R} \quad (44)$$

$$X_p' = \frac{x_p}{1 + S_c}\left[1 + \frac{S_c}{K_c - x_p(K_c - 1)}\right] \quad (45)$$

$$G_1 = G + L_1(S_c - S_h) \quad (46)$$

$$x_d = \frac{G_1(1 + H_c)X_p'}{L_1(1 + S_c)\beta_c} + \frac{L_1 x_c(1 + S_h) - GY_c(1 + H_c)}{L_1(1 + S_c)} \quad (47)$$

$$m_c = \frac{L_1(1 + S_c)}{G_1(1 + H_c)} \quad (48)$$

$$b_c = \frac{GY_c(1 + H_c) - L_1 x_c(1 + S_h)}{G_1(1 + H_c)} \quad (49)$$

$$N_c = \left[\log\frac{(\beta_c m_c - 1)x_d + \beta_c b_c}{(\beta_c m_c - 1)x_c + \beta_c b_c}\right]\frac{1}{\log \beta_c m_c} \quad (50)$$

$$X_t' = \frac{L_1 x_d(1 - H_h S_c) + G_1 X_p'(H_h - H_c) - Px_p(1 - H_h S_c)}{L_1(1 - H_h S_c) + G_1(H_h - H_c) - P(1 - H_h S_c)} \quad (51)$$

$$X_t = \frac{X_t'\beta_h(1 + S_c)}{\beta_h(1 + S_h)(1 - H_h S_c) + (S_c - S_h)(1 + H_h)} \quad (52)$$

$$G_2 = G + \frac{(S_c - S_h)(L_1 - P) + GH_h S_c}{1 - H_h S_c} \quad (53)$$

$$m_h = \frac{L_4(1 + S_h)}{G(1 + H_h)} \quad (54)$$

$$b_h = Y_0 - m_h x_h \quad (55)$$

$$N_h = \left[\log\frac{(1 - \beta_h m_h)x_h - \beta_h b_h}{(1 - \beta_h m_h)X_t - \beta_h b_h}\right]\frac{1}{\log \beta_h m_h} \quad (56)$$

Subject to

$$0 < R < 1 \quad (57)$$

$$0 < e_R < 1 \quad (58)$$

$$0 < e_h < 1 \quad (59)$$

$$0 < G < 500 \quad (60)$$

$$G \div \left\{ \frac{(1 + S_c)(x_d - X_c)}{(1 + H_c)\left(\frac{x_d}{\beta_c} - Y_c\right)} - (S_c - S_h) \right\} < L_1 < G \div$$
$$\div \left\{ \frac{(1 + S_h)(x_t - x_h)(1 - H_h S_c)}{(1 + H_h)\left(\frac{X_t}{\beta_h} - Y_0\right) - (H_h - H_c)(1 + S_h)(X_t - x_h)} \right\} \quad (61)$$

$$0 < X_c < \beta_c Y_c \quad (62)$$

$$0 < L < G \div$$
$$\div \left\{ \frac{\beta_c e_h e_R (1 + H_h)^2 (1 - R)}{R\beta_h^2 (1 + S_c)(1 + H_c) + R x_f \beta_h^2 \beta_c (1 + S_h)(H_h - H_c)} - \frac{e_R e_h (1 + H_h)(1 - R)}{\beta_h R (1 + S_h)} \right\} \quad (63)$$

8. References

1. Booth, A. D. (1957). *Numerical Methods*, Butterworths, pp. 95–100, 154–160.
2. Davies, O. L. (Editor) (1956). *The Design and Analysis of Industrial Experiments*, Oliver & Boyd, pp. 495–578.
3. Booth, A. D. (1949). "An Application of the Method of Steepest Descents to the solution of Systems of Non-linear Simultaneous Equations," *Quart. Journ. Mech. and Applied Math.*, Vol. 11, Part 4, pp. 460–8.
4. Box, G. E. P., and Coutie, G. A. (1956). "Application of Digital Computers in the Exploration of Functional Relationships," *Proc. I.E.E.*, Vol. 103, Part B, pp. 100–7.
5. Brooker, R. A. (1958). "The Autocode Programs developed for the Manchester University Computers," *Computer Journal*, Vol. 1, pp. 15–21. "Further Autocode Facilities for the Manchester (Mercury) Computer," *ibid.*, pp. 124–7.

Correspondence (*continued from p.* 174)

If the data is not random, but contains long strings of consecutive items, the merge-sorting time is reduced somewhat, whereas with the tree-sorting method it is substantially increased. In the worst case, if all the data is in order, a merge-sort discovers this fact in one pass, whereas a tree sort takes $O(N^2)$ steps [instead of $O(N \log N)$ steps] to deal with the data. This has turned out to be a serious problem in practice. There are two ways round the difficulty. One is to modify the tree process to prevent the growth of excessively long branches going continuously to the right or left. Lane of I.C.I. has done this successfully in a statistical program. The other way is to use a random-access procedure for storing the data, and then to sort the randomly-stored data before output.

With the random access method, the sorting can be avoided altogether if an index of all possible keywords already exists. The procedure is as follows. During input, a pseudo-random function of the keyword is used to determine an address in a strip of locations from which to start searching for a vacant place in which to put the next keyword. The keyword is put in the first available location from this point onwards, treating the strip as joined end to end. During output, the successive possible keywords are read in from the index and the corresponding starting address on the drum is generated by the pseudo-random function. The strip is searched from this point onwards, until either the required keyword is found, or until a vacant location is found. In the first case the required data is output, in the second case the next keyword from the index is input.

If, however, no index is available, the tree-sorting method may be used to sort the data stored by the random access procedure. As this procedure will have broken up any long strips in the data, the tree-sorting method will be efficient.

The time taken by the random access procedure depends on the emptiness of the storage strip. If N items with different keywords are accessed, the mean number of store accesses for each keyword may be shown to be about:

$$\frac{S}{N} \log_e \frac{(S)}{(S - N)}$$

where S is the number of storage locations available. It is possible to arrange to refer to the keywords indirectly through an index of addresses. As the addresses will occupy only a fraction of a word (a third on Pegasus) it is quite easy to keep the effective number S large compared with N. If, for example, the keywords are referred to by three 13 bit tags in a 39 bit Pegasus word, and if there is one tag word for each keyword, then at worst N cannot exceed $\frac{1}{3} S$. The result is that at worst about $2\frac{1}{2}$ accesses per item are involved, including access to the index.

If the data consists of one item per keyword, then the normal tree-sorting time computed by Windley must be added to this involving

$$1{\cdot}4\,(N + 1) \log_2 N + (1 + \;)N \text{ accesses.}$$

If, however, as is frequently the case, one wants to accumulate totals against keywords (e.g. costs against cost collection numbers), then there is a substantial economy as the result of random accessing the data and only tree-sorting the totals. If T is the number of totals into which the N items fall, the number of accesses will be

$$2{\cdot}5\,N + 1{\cdot}4\,(T + 1) \log_2 T + (1 + \gamma)T$$

as compared with $\quad 1{\cdot}4\,(N + 1) \log_2 N + (1 + \gamma)N$

for the number of accesses if the raw (consumed) data is tree-sorted on input. If an index is available, the number of accesses becomes

$$2{\cdot}5\,(N + T)$$

In practice, in either case, the random access time arising from N will usually dominate the analysis.

C. M. Berners Lee

Author's reply

If the keys are very long (several complete words, say) then a merging method cannot use block transfers. If a key and an address can be packed into one word, then I agree that merging is the process to be used on a machine with a magnetic drum main store with block transfer facilities to a high-speed store. It is, of course, well known that the tree method is slow if the data already contain long strings of ordered keys before sorting commences, and it should not be used in these cases.

P. F. Windley