A method for curve seeking from scattered points

By P. H. A. Sneath*

The paper describes a computer method for seeking curves through sets of scattered points in many dimensions by adjusting the positions of the points to form smoothed curves. The sequences of the points are obtained, and linear, branched and closed sequences can be found. The method is applicable to many problems involving trends in characters or time in biology, medicine and other fields.

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

1. Introduction

One of the pressing needs of numerical taxonomy (i.e. numerical methods of biological classification) is a method for finding trends or sequences in biological data. Such trends are easily found intuitively when the data are simple and when only one or two well-separated trends are present, as represented by sequences of objects that fall along smooth curves. They are more difficult to find when sequences are complex (e.g. branched), or when the characters of each object are numerous. An example of a simple sequence is the trend produced by growth of an individual; here a scatter diagram of age against weight (or against many other variables) will readily reveal the growth curve. A recent article by Fry (1964) raises the problem of complex trends in an acute form. He was working with marine animals (pycnogonids), where the many available characters of the animals change in a complex manner with age. In addition several different but very similar species were present in the samples under study. However, the usual methods of numerical taxonomy (see Sokal and Sneath, 1963) tended to group together the young specimens of all the species, while the oldest specimens of all the species also grouped together. In fact, Fry's material contained several separate and roughly parallel trends, one for each species. What was needed was a way of distinguishing these from the specimens alone, since direct observation of the growth of the living animals was not practicable. It is a familiar difficulty in creatures showing considerable steady developmental changes, such as grasshoppers. Here the sequence forks into the two sexes. The position can be still more complex. In termites, for example, the newly hatched nymphs are all alike, but they develop into different adult forms: workers, soldiers, queens, etc. Very similar problems occur in other fields, such as the progression of symptoms in diseases, or the evolutionary pathways in recent and fossil material [e.g. Boyce (1964); Edwards and Cavalli–Sforza (1964)].

Most methods are only adapted to single unbranched sequences (except for some parts of graph theory, see Rose, 1964). Single straight lines can be fitted by regression analysis, and curvilinear regression can fit quadratic, cubic and higher order curves. If some variable is independent (such as time in a temporal sequence) one can fit the curve easily but roughly by simpler procedures (such as taking the mean of the characters at successive time periods). But these methods break down with forked or recursive trends. The single-link clustering method (Sokal and Sneath, 1963, p. 180) is in theory capable of picking out long parallel chains. Yet when applied to some archaeological material in which a sequence of cultural traits was present (Hodson, Sneath and Doran, to be published) it was not satisfactory, since it may bring together different ends of the same sequence. It is also not adapted to forked sequences. A modification of single-link analysis has been used here (see Section 4) which is related to graph theory. By this the nearest neighbour to a point is found, then the point nearest to the second, and so on. This gives the sequence separated by the smallest distances, but difficulties arise with branched sequences in the simplest form of this modified single-link analysis. A more elaborate procedure has therefore been developed.

If one had an infinitely finely-graded sequence of points it would be easy to use the ordinary mathematical concept of a continuous curve. In practice we have gaps between the points and also some variation about the ideal curve due to the variability of the entities which are studied. What is in theory a smooth curve is in practice a swathe of scattered points.

This paper describes a "gravitational" model in which the analogue of the "gravitational force" is used to move the scattered points from the swathe to the axis of the swathe, thus giving a smooth curve. The programming is not yet completed (an ALGOL program is being written for the Atlas, since a large computer is needed) but the technique is published here for those who may wish to elaborate it further. **Fig. 1** shows a two-dimensional example with which the reader can follow the steps of the method. Clearly there is a forked trend present, with one limb directed to the left and other directed downward but also kinked. Small offshoots are also present, and a few outlying points which are not obviously on the trends. For convenience the description is broken into numbered sections and arrays are labelled as in the ALGOL convention.

* *Medical Research Council Microbial Systematics Research Unit, University of Leicester, University Road, Leicester.*

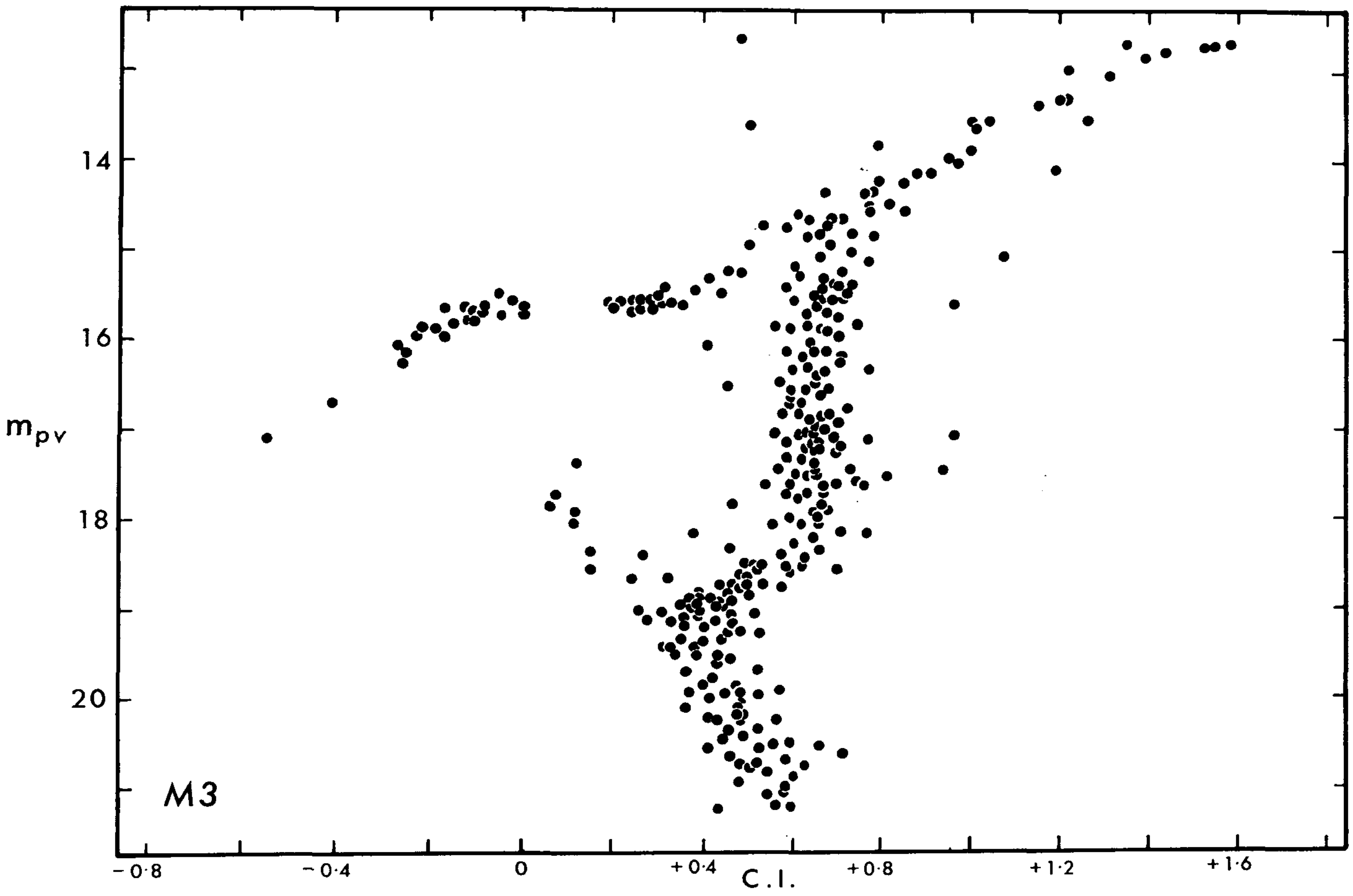

Fig. 1.—Selected points from the Hertzsprung–Russell diagram of the stars of globular cluster M3 adapted from a figure in Bizony (1964, p. 135). The ordinates are Colour Index and Apparent Magnitude. The Main Sequence of stars runs from bottom to top left, and is poorly represented in the upper part. The Hertzsprung Gap is shown near the left of the trend which runs from the left toward the top right.

2. The collapsing process

The collapsing process (to move the points into curves) is controlled by preset parameters:

2.1. Maximum movement permitted, M. The value of M is the maximum movement that is allowed for any point during any one cycle. It controls the fineness of the collapsing process. If too large, all points will collapse to the centroid of the system in one cycle and this can also lead to undesirable to-and-fro movement in later cycles. If M is too small the running time becomes too long. However, M is reduced by linearity (current value for a point is Z, see 2.5 below), and finally becomes very small for most points, which causes exit from the collapsing process since the points then cease to move.

2.2. Size of a neighbourhood, R. The value of R is the radius which defines the neighbourhood of a point. Any other point within R of a point i is in the neighbourhood of i. R determines the width of a swathe which will collapse into a curve. If R is too small every point will be a "lone point" without a neighbour, and no sequences will emerge. If too large, all points will form one line and the method will lack discrimination. In general two swathes will not be recognized as distinct if they are within R of each other.

2.3. Criterion of linearity, L. No *a priori* decision can be made about whether a curve will be linear or of higher order. Therefore the curves are determined from short sections of length $2R$ which are made as linear as is feasible. The criterion of linearity is L (observed values are Y). This determines how close to a straight line must be the other points in the neighbourhood of a given point i before point i is "frozen" in its current position. This happens when Y_i equals L. If L is too small (poor linearity is permitted) the swathe remains diffuse and collapse is slow. If L is too large then all

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

points are forced into a single straight line, since we are too exacting about linearity, and all curves are straightened and finally pressed together.

2.4. The useful values of M, R, and L are discussed in Section 5 since the factors affecting their choice can be best seen from the next few sections.

2.5. Input of data. The input consists of a matrix recording the values of n characters of t entities, (with reference numbers from 1 to t). This forms array $A[1{:}t, 1{:}n]$. Provision could perhaps be made for missing data, but this is not considered here. The characters should be suitably scaled, either into a fraction of the maximum range (giving values between 0 and 1) or else standardized with each character having a mean of zero and unit variance (giving values in practice between about -3 and $+3$). Unscaled characters could be used if the ranges were not too dissimilar. The method is based on a model of continuously varying characters, but could perhaps work with 1, 0, characters. Each entity is represented by a point in n-dimensional Euclidean space, so that the positions of the t points are obtained by treating the values of the n characters as the n co-ordinates, given for any point the characters with values $x_1, x_2, x_u \ldots x_n$.

If these points are scattered in swathes then "gravitation" would cause them first to collect into curved sequences; this is achieved by making the "gravitation" inversely proportional to the square of the distance, thus first pulling together the closest points. Secondly, the curves would shorten and collapse into one point at the centroid of the system. The second collapse is checked by "freezing" the points (by reducing M for each point) when a neighbourhood becomes acceptably linear (by the criterion L). The maximum movement allowed, therefore, differs from one point to another, and for point i is symbolized by Z_i. This preserves the curved sequences.

2.5.1. The collapse will reduce to some extent the overall scatter, thus reducing the scale of the whole system. This reduction can be recorded by estimating the total variance about the common mean point $\bar{x}_1, \bar{x}_2 \ldots \bar{x}_n$, whose coordinates are preserved in Array $S[1{:}n]$ since they are required later (2.13). This is therefore estimated as the Original Variance,

$$\sigma^2_{\text{original}} = \frac{1}{t} \sum_{u=1}^{u=n} \sum_{i=1}^{i=t} (x_{iu} - \bar{x}_u)^2$$

The Final Variance is estimated at the end of the collapsing process (see 2.12).

2.6. Calculation of force acting on a point i. The "force" acting on a point i (commencing with $i = 1$) is now calculated. Point i is compared in turn with all the rest. The array $B[1{:}n]$ holds the current components of the force in the n dimensions, and this is initially filled with zeros. The array $C[1{:}t]$ contains the current value of Z_i for each point i $(i = 1, 2, \ldots t)$, and is initially filled with the preset value M. If Z_i becomes zero then point i cannot move, and the next point is taken.

2.6.1. The force acting on point i from point j is called F_{ij} and it is set equal to $\frac{K}{d_{ij}^2}$ where d_{ij}^2 is the total squared distance between i and j, and K is a constant which is here taken as unity, but may be adjusted if required. Then

$$d_{ij}^2 = \sum_{u=1}^{u=n} (x_{iu} - x_{ju})^2.$$

2.6.2. In order to partition the total force F_{ij} into the components in the n dimensions one calculates the force due to dimension u as

$$F_{iju} = \frac{Kd_{iju}}{|d_{ij}|^3} \quad (u = 1, 2, \ldots n;\ i \neq j).$$

Having calculated the uth component of the force on i due to j, one adds all such values $(j = 1, 2, \ldots t; j \neq i)$ into array B. Thus $B[u]$ contains the value of the uth component, F_{iu}, of the total force vector acting on i. The absolute value, F_i, of this is required:

$$F_i = \sqrt{\left\{\sum_{u=1}^{u=n} (B[u])^2\right\}}.$$

2.7.1. Next one tests if F_i is greater than Z_i (where Z_i in location $C[i]$ is the current value of the maximum movement permitted to i. If $F_i \leqslant Z_i$ then point i is "moved" by the distance F_i, that is it is "moved" in the n dimensions by a distance equal, for dimension u, to F_{iu}. This is done by adding algebraically the value of F_{iu} to the value in array $A[i, u]$, and so, in turn, $F_{i1}, F_{i2}, \ldots F_{in}$ to locations $A[i, 1], A[i, 2] \ldots A[i, n]$. However, if $F_i > Z_i$ then point i is moved only by a proportionately smaller amount, $(F_{iu}Z_i)/F_i$ for dimension u. It is probably immaterial that the distance moved is abruptly checked at Z_i; a smoother relation from an asymptotic function (such as arctangent) could be used.

2.8. Having moved point 1, the next step is to move points 2, 3, ... t by returning to 2.6.2. It is true that having moved the points previously considered the system is not in its original state, but with a sufficiently small value of M this is immaterial, and it also obviates seesawing of the points, which could otherwise be slow to cease.

2.9. All points have now been moved one step, and a second cycle of movement can be started. One must now discover if the movement has been sufficient to give the required curves. This may occur more quickly in some parts than in others where the swathes are more diffuse. One needs therefore a test of linearity over small distances, so that areas where collapse is sufficient will have their points "frozen." For this the neighbourhood of each point (Section 2.2) is examined for linearity. Since the criteria of linearity are varied, discussion of a suitable criterion is postponed to Section 3. The observed linearity, Y_i of the neighbourhood of point i is a quantity varying from 0 (for non-linearity) to 1 (for all points in the neighbourhood of i lying on a straight line). The points in the neighbourhood of i are found

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

using d^2 (see 2.6.1) and determining if d^2 is less than R^2 (Section 2.2).

2.10. If Y_i is greater than the preset criterion L (Section 2.3) then linearity is acceptable, and Z_i is reduced to zero, thus preventing any more movement of point i. If $Y_i \leqslant L$, then replace Z_i by $M(1 - Y_i^2)$. Other functions of Y could be used, but this allows free movement when linearity is poor and drastically reduces it as linearity becomes high. The function allows a value of Z_i which was previously zero to become positive again if required by the point configuration, but this carries some risk of remobilizing frozen areas to an undesirable extent. If so, then Z_i can be replaced by $M(1 - Y^2)$ only if $Z_i > 0$; otherwise Z_i remains zero. This second alternative carries the danger that near a fork the points at the branch will be pulled to the ends of the limbs of the fork, since the latter, being linear, will be irreversibly frozen, and hence the sequence will be broken by the appearance of a gap larger than R. The first alternative seems preferable, therefore, though it would allow the slow collapse of the arms of a Y-shaped sequence.

2.10.1. All points in turn are tested for linearity and the Z values in array C are adjusted.

2.11. All t points have been tested for linearity, but several cycles of movement will be needed to reduce the swathes to curves. The cycles are entered at 2.6.1 and repeated until linearity is acceptable, but not too stringent. Non-linearity is due to curves and also to branches (since points near the fork must also have poorly linear neighbourhoods). If exit from the cycles is too delayed, the curves may be stable (because Z_i has been reduced to zero) but the points near the fork may move enough to break the sequence (which occurs if a gap appears that is more than R). It seems possible that points near forks may be rather unstable, and so it would be safer to stop the collapsing fairly early, despite the danger that some very diffuse parts of the system may not have collapsed sufficiently. The degree of total movement in a cycle might be an adjunct to controlling this, since when this became small it would indicate that collapse was nearing completion. It could be obtained by summing F_i.

2.11.1. One can therefore test $\frac{1}{t}\sum_{i=1}^{i=t}(1 - Z_i^2) > L$. If not, another cycle is entered at 2.6.1; otherwise one proceeds to 2.12.

2.12. The collapsing process being completed, the points should now lie in sequences which are linear over short distances but can be curved or branched on a larger scale. The overall contraction of the system is now found by estimating the Final Variance σ^2_{final} (see 2.5.1) from the final x values.

2.13. The degree of collapse from the original configuration to the curves that have been found may be measured by estimating the variance of the difference between the original and final positions of the points:

$$\sigma^2_{(\text{original}-\text{final})} = \sigma^2_{\text{original}} - \sigma^2_{\text{final}}$$

where:

$$\sigma^2_{\text{final}} = \frac{1}{t}\sum_{u=1}^{u=n}\sum_{i=1}^{i=t}(x_{iu\,\text{final}} - \bar{x}_{u\,\text{original}})^2.$$

The original means are in array S. This statistic cannot offer an absolute criterion for successful collapse since there is no absolute criterion for the final curves.

Section 3 discusses criteria of linearity, so the next program step is at 4.2.

3. Criteria of linearity

The test of linearity over small distances must be quick, simple and sensitive to small departures from colinearity. A convenient test uses the relation between the sides of a triangle: the sum of the two shorter sides is greater than the longest side. The excess is a measure of linearity, and if it is zero the three vertices are colinear.

For a point i the other W points in its neighbourhood (i.e. within R) are found; the most distant pair of points is then determined by searching the matrix of interpoint distances of the neighbourhood i for the largest value. These two points may be called a and b (i can itself be one of them, but usually is not). Then for all W points in the neighbourhood calculate:

$$Y = 1 - \frac{1}{Wd_{ab}}\sum_{i=1}^{i=W}(d_{ai} + d_{ib} - d_{ab}).$$

This coefficient thus represents one minus the proportionate average excess distance in travelling from a to b via the other points instead of direct. It lies between 0 (e.g. for an equilateral triangle) and 1 for complete colinearity. For points evenly spaced within a circle it is about 0·84, and somewhat less for points in a hypersphere of many dimensions.

This coefficient has also a useful property: for a given perpendicular distance h from the line ab it is higher when i is equidistant from a and b than when it is close to one or the other (equilinear points lie on an ellipse with foci at a and b). This allows an obtuse triangle to have a better linearity score than an acute one, and hence facilitates fitting of smooth curves to the points. For Y close to 1, if i is equidistant Y approaches $1 - (2h^2/d_{ab})$, and when i is close to a or b, Y approaches $1 - (h/d_{ab})$.

The perpendicular distance h is a more sensitive test of departure from linearity. It can be found by dividing twice the area of the triangle by d_{ab}. The area is obtained from the usual formula, as $\sqrt{[s(s - d_{ab})(s - d_{ai})(s - d_{ib})]}$ where $s = \frac{1}{2}(d_{ab} + d_{ai} + d_{ib})$. It requires a good deal more computation, however.

If the neighbourhood of i contains no other point, or just one, then Y is indeterminate, and is taken as 1. This serves to anchor down isolated points, isolated pairs and free ends of sequences, but if this proves too rigid Y can be taken as a number less than 1.

4. Finding the sequences

Finding the sequences is the most involved procedure in the process, because of the many topological alternatives.

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

4.1. There is no difficulty in output of the final positions of the points; their co-ordinates are given by printing the final values of array A. These are almost equivalent to the projection of the points onto the curves representing the idealized sequences, and thus bear some resemblance to loadings in factor analysis. But this table is not a handy summary and the sequences of the points with their reference numbers are also needed.

4.2. Owing to the complexity of the topology the sequences are assembled in unbranched sections, called *limbs*. A limb terminates either (*a*) as a free end, or (*b*) at a fork in the sequence, or (*c*) when the starting point is reached because the sequence is a closed loop. The limbs are assembled as far as possible into complete sequences, but branched sequences are printed as limbs together with an indication of the identity of the branch point and the symbol B to draw attention to it. Thus the sequences 1, 2, 3, B4; 20, 21, 22, B4; 16, 15, 14, B4, would be readily seen to form a Y-shaped sequence with point 4 at the junction. The three arms are treated as separate sequences in the search process and are thus referred to, though they should properly be called subsequences (but not limbs; an unbranched sequence has two limbs).

A point can be of several kinds with respect to the topology. With their symbols they are: "sequence starters" (S), "limb enders" (E), "branch points" (B), "lone points" (SE), "sequence closer" (C) and "chain points" (no symbol).

4.2.1. A sequence starter, S, is the point at which the search of a limb is begun. It can be deleted from the system when its sequence is complete. It may also be in practice a limb ender if the sequence is a closed curve. When the first limb ends, the search is recommenced at S in order to find the other limb (except for closed sequences).

4.2.2. A limb ender (E) is the point at a free end of a sequence, such that no other point (which has not been found previously in the search) lies in its neighbourhood, i.e. within a distance of R. Branch points (B) are also in practice limb enders, but are treated separately. A limb ender can be deleted from the system when its limb is complete.

4.2.3. A branch point, B, is one which has a non-linear neighbourhood, as shown by $Y < L$. At the fork there will probably be several points of this kind, since there will be some overlap between adjacent neighbourhoods. A branch point terminates a limb and leads to a secondary search for contiguous branch points in order to identify unambiguously the fork in question. Branch points are never deleted.

4.2.4. A lone point is marked SE since it both starts and ends a sequence of one, and can then be deleted from the system. It is a point with no other point within a distance of R.

4.2.5. A sequence closer (C) is found in a closed sequence, and is the sequence starter if this is found a second time.

4.2.6. Points in the middle of limbs which have linear neighbourhoods are chain points. They are not marked when printed. They can be deleted when their sequence is complete.

4.2.7. In order to keep a tally of these different kinds of point, four arrays are used: array $G[1:t]$ which records the status of the points in the current sequence; $H[1:t]$ which records deletions and branch points; $P[1:t]$, which records contiguous branch points; and $Q[1:t+1]$ which records the reference numbers of the points in the current sequence.

4.3. The array H is filled with $+1$ values. Then for each point i (from 1 to t in turn) the neighbourhood is tested to see whether it is empty or non-linear ($Y_i < L$).

4.3.1. Points with empty neighbourhoods are lone points and are printed in the form S i E, while $H[i]$ is set to -1 to indicate deletion of point i.

4.3.2. If the neighbourhood is non-linear the point is printed with the symbol B (as a check list) and $H[i]$ is set to 0.

4.4. The array Q is filled with zeros and arrays G and P are filled with $+1$ values. This becomes the start of compiling a sequence.

4.5. The first point in array H that is marked $+1$ is used as the sequence starter (which cannot be a lone point, any other deleted point, or a branch point). This point, j, is marked as S by making $G[j]$ zero, and its reference number j is placed in $Q[1]$.

4.6. Point j is now used as the centre of a neighbourhood. All the other $t-1$ points are first checked in array H; if they are marked -1 they have been deleted and the next point is considered. If all other points are -1 the sequences have all been found (go to 4.12). The remaining points marked $+1$ or 0 are now tested to see if they are within R of j and if so the distances of these v points are noted.

4.6.1. Each of the v points is now checked in array G to confirm that it is marked $+1$. If not, it has been used previously in the sequence, and the next point is considered. If none of the v points are $+1$, then there is no unused point available and the limb ends (go to 4.8). The nearest of the v points which are marked $+1$ is then found; this is point k.

4.6.2. Point k is the nearest available point to j, but it may be a branch point. $H[k]$ is therefore tested, and if zero, go to 4.7 for the special subroutine for bıanch points.

4.6.3. If $H[k]$ is $+1$, place the reference number of k in $Q[2]$, and set $G[k]$ to -1. The search for the nearest neighbour to k is continued by re-entering at 4.6 using k as the centre of the neighbourhood.

4.6.4. Successive nearest neighbours are thus noted in $Q[1]$, $Q[2]$, $Q[3]$. . . .

4.7. The nearest neighbour is a branch point, and this terminates the limb. All contiguous branch points are now required. The neighbourhood of k is searched for the nearest point which is marked 0 in array H, and $+1$ in array P. Its reference number is recorded in the next available location of array Q, while in P it is marked zero to prevent it being found a second time. The

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

successive nearest neighbours which are branch points are thus obtained. When no more can be found in the last neighbourhood the limb is ended (go to 4.9).

4.8. No available point can be found in the neighbourhood of the last point in the limb. The limb has come to an end for one of two reasons:

4.8.1. The nearest point which is marked +1 in H is tested to see if it is marked 0 in G and is also within a distance of R. If so, it is the sequence starter (4.5), to which the sequence has led back. The sequence is therefore a closed loop, and it is complete. It is printed out commencing with the symbol S and the reference number of the sequence starter, and ending with this number repeated and the symbol C to indicate closure. The sequence is available in correct order in array Q from $Q[1]$ to the first location containing a zero. The program goes on at 4.12.

4.8.2. If the conditions in 4.8.1 are not satisfied the limb has a free end. It is now necessary to find the other limb.

4.9. The other limb is found by returning to the sequence starter (0 in array G), and entering the program at 4.6. This is followed as before except that in 4.6 and 4.7 the reference numbers are entered in locations $Q[t + 1]$, $Q[t]$, $Q[t - 1]$ . . . etc. The use of $t + 1$ locations ensures that at least one remains zero and marks the ends of the limbs. On return to 4.9 for the second time the program leads to 4.10.

4.10. The array Q now holds the complete sequence in two sections, and it is printed in the order $Q[t + 1]$, $Q[t]Q[t - 1]$ . . . until a zero is found, and then $Q[1]$, $Q[2]$ . . . until a zero is found again.

4.11. At print-out the reference numbers of the points are printed with the appropriate symbols by examining array H for branch points. The first and last in the sequence are marked E if they are not marked B. If required the Euclidean distance between successive members of the sequence can also be printed (calculated from array A) together with the cumulative sum. This would assist study of the spacing and linearity of the sequence.

4.12. The points in the sequence are then marked -1 in array H except for branch points which remain as zero. Then the program returns to 4.4 if any points in H are still marked +1. If not, the program is completed.

5. Choice of preset constants; considerations affecting their size

Experience will be needed before firm advice can be given on the best values to choose for the preset constants M, R, and L. They will clearly depend in part on the number of points. A few points cannot give a very significant sequence. In particular R must be fairly large or no sequences will appear.

5.1. It would seem reasonable for R to be so chosen that generally several points fall into one neighbourhood, and an appropriate value would be to make a rough estimate of the average range of the characters and choose for R a value which included 3–5 points (assuming points to be scattered more or less randomly through the ranges). For standardized characters R of $10/t$ to $20/t$ would seem reasonable.

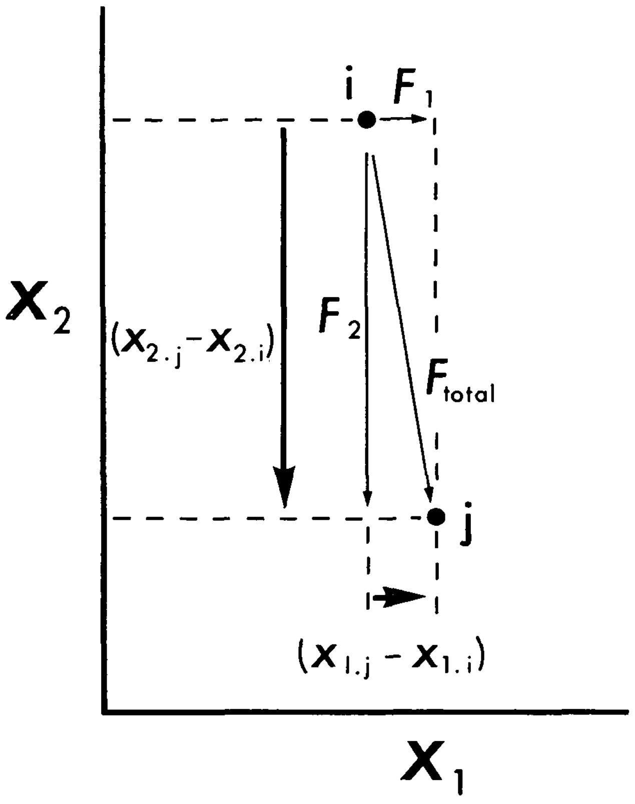

Fig. 2.—Forces acting on a point i from another point j for two dimensions, 1 and 2.

The total force is inversely proportional to the square of the distance between i and j. It is partitioned into a component in dimension 1, F_1, which is proportional to the distance between i and j in dimension 1, i.e. $(x_{j1} - x_{i1})$ and into the component F_2 in dimension 2, proportional to $(x_{j2} - x_{i2})$.

5.2. The value for M depends largely on the computer time that is readily available. Probably at least ten clustering cycles are needed, which suggests an upper limit for M of about 0·02 of the average range, or about 0·1 for standardized characters (assuming swathes about 1/5 the width of the whole system).

5.3. Suitable values of L are less easy to suggest. If L is too high any curves will be forced into straight lines, and forked sequences will have their limbs brought together and fused. The choice of L also depends on R, for if R is increased more points will in general come into a neighbourhood, and linearity will decrease. Moderate non-linearity to permit gentle curves can be allowed by taking the curvature which would permit one circular sequence to fit comfortably into the character space. If 3–6 points are to be in each neighbourhood this would give a neighbourhood that subtended an arc of about $10/t$ radians, and the circle could be allowed a radius of $\frac{1}{4}$ of the average range (or for

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

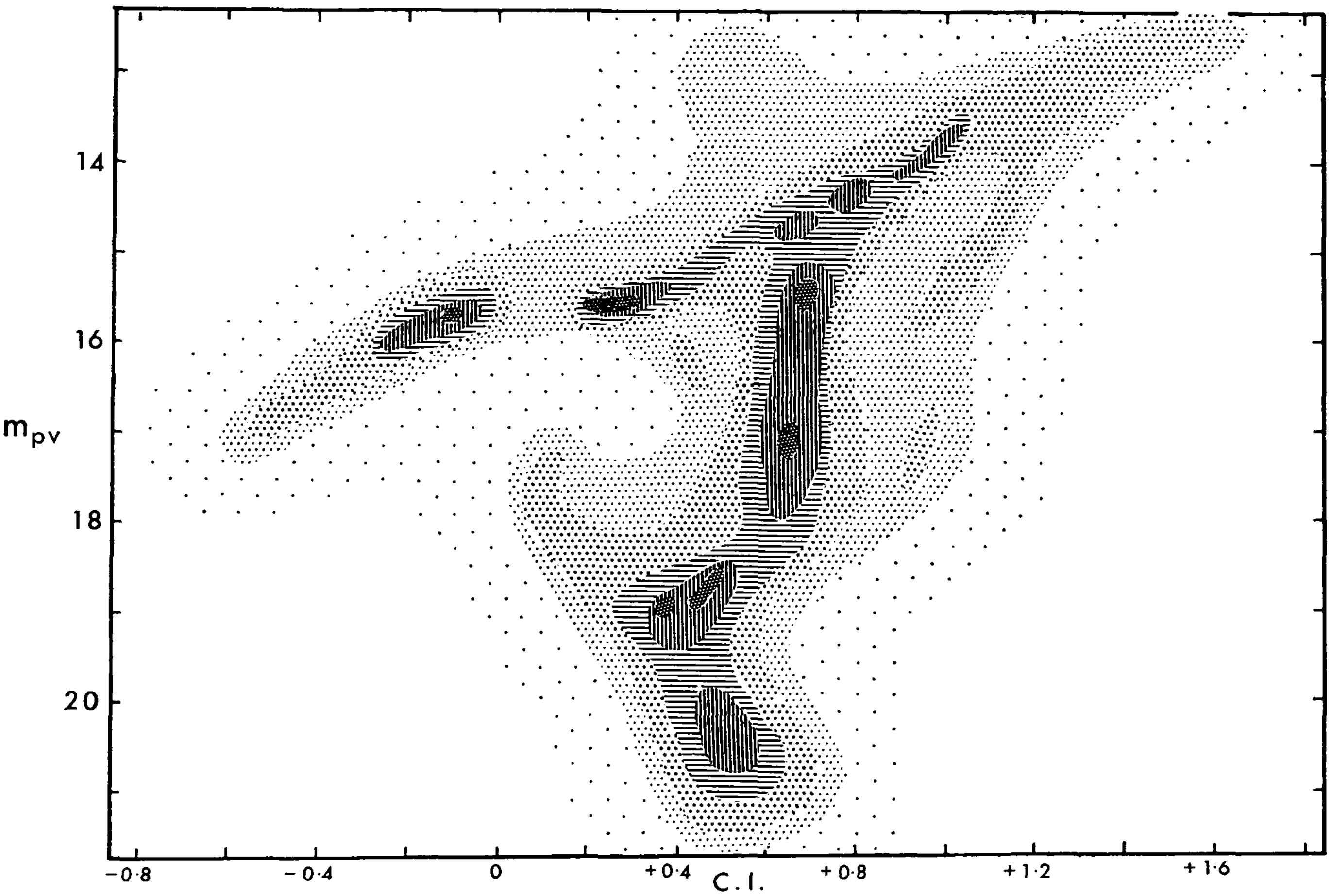

Fig. 3.—Representation of Fig. 1 as a density diagram with the discrete points replaced by a field. This is very similar to a poorly-focused photograph of Fig. 1

standardized characters a radius of 1). For points on an arc of θ radians where θ is small, Y has a value of about $1 - (\theta^2/32)$ for three equally spaced points, and about $1 - (\theta^2/38{\cdot}4)$ for five equally spaced points. An arc of $10/t$ radians would require L of about $1 - (3/t^2)$ to allow the desired degree of curvature. This could be used as a guide to the upper limit of L.

6. Discussion

The procedure described here is undeniably complex. Indeed it is surprising how difficult it is to imitate on a computer a process which is swiftly accomplished by the eye whenever the data can be presented graphically in a suitable form (e.g.). **Fig. 4.** Yet a consideration of the mental steps involved suggests that they are similar in many ways to the process given here. One does "move" points mentally, and fit curves through swathes, and follow sequences first one way and then back the other way. No very great simplification has come about from pondering on the problem. One alternative would be to replace each point by a field, which would lead to bands of force replacing the swathes, and which can be mimicked by blurring the points in Fig. 1 (as, for example, putting the picture out of focus). **Fig. 3** shows a density diagram of this kind based on Fig. 1. But one is still faced with the problem of tracing these bands through multidimensional space, and whether one used an envelope or contour method (such as helical searching round limb along surfaces of equal density) or fitted lines through the bands, it seems unlikely that the process would be much simpler to that given here. It is much to be hoped that simplification and improvement can be made in the future.

The "inverse square law" of gravitation may not be the best, and it would be interesting to try in Section 2.6 formulae where $F = K/d^{\lambda}$, with λ other than 2. The analogy with three-dimensional space would require λ to be $n - 1$. However, the main effect of changing λ will probably be to change the effective range of the attractive force, with high λ giving close-range forces. Another interesting development would be to modify the method to find surfaces instead of curves. This would require substituting a criterion of planeness for the criterion of linearity (the analogue of the criterion used here would be the relation between the areas of the faces of a tetrahedron). This might have applications in fields

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

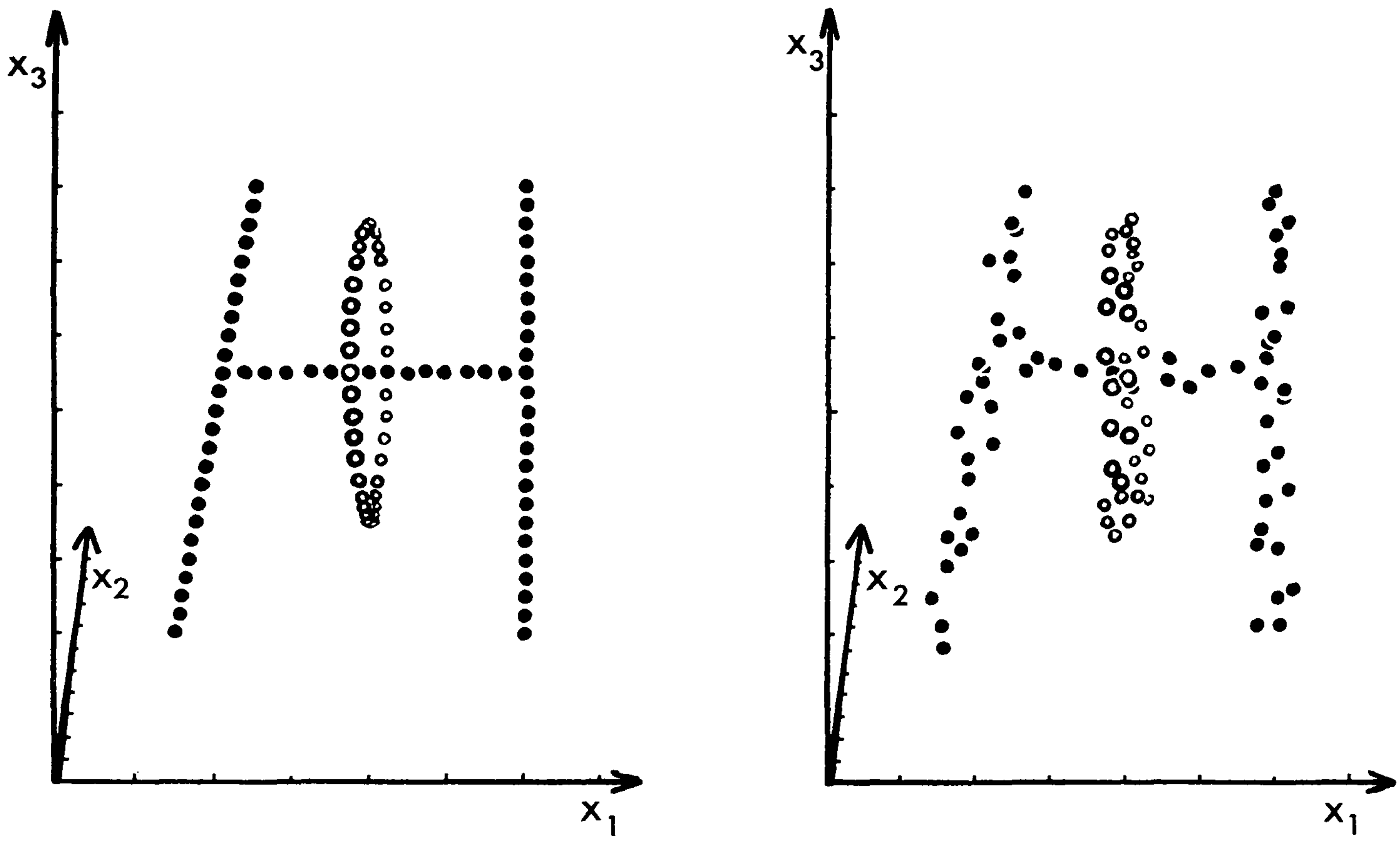

Fig. 4.—Illustration of complex curve-seeking. On the left is a figure in three dimensions, x_1, x_2, x_3, consisting of points forming an H with an O around the crossbar. On the right is the same figure with the points randomly perturbed a small amount. The hope is that from the latter it will be possible to recover the original configuration together with the sequences that describe the topology

such as geology. Although it is not possible to postulate *a priori* the algebraic order of the curves, so that the usual criteria of goodness-of-fit are not available, it would be possible to calculate the least-squares goodness-of-fit of the empirically observed line segments which join successive members of the sequences, and this would provide some test of the adequacy of the collapsing under different conditions.

There are a number of points which cannot be satisfactorily cleared up until some examples have been tried. One of these is whether within a diffuse cloud of points one might get several spurious curves during collapse. It seems possible that chance aggregations might collect more points about themselves and lead to this effect. Another problem is the instability of branch points, which has been mentioned, but which seems difficult to study theoretically.

A further development would be to reduce the dimensionality of the final positions to two (by a process such as principal component analysis with retention of only the first two vectors). One could then print out a suitable diagram of the position of the points in two dimensions, which would allow the advantages of visual study.

Acknowledgements

I am grateful to M. J. Sackin and J. Doran for criticism and mathematical advice, and to Miss Lesley Muir for assistance with programming.

References

Bizony, M. T. (editor) (1964). *The Space Encyclopaedia: a Guide to Astronomy and Space Research.* Horsham, Sussex, England: Riband Books (paper-back, 288 pp., illust., republication, Artemis Press, 1957).

Boyce, A. J. (1964). "The Value of some Methods of Numerical Taxonomy with reference to Hominoid Classification." Pp. 47–65 in *Phenetic and Phylogenetic Classification.* Systematics Association Publication No. 6 (London: Systematics Association, xi + 164 pp., illust.).

(*References continued overleaf*)

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024

EDWARDS, A. W. F. and CAVALLI-SFORZA, L. L. (1964). Reconstruction of Evolutionary Trees. Pp. 67–76 in *Phenetic and Phylogenetic Classification.* Systematics Association Publication No. 6. (London: Systematics Association, xi + 164 pp. illust.).
FRY, W. G. (1964). "The Pycnogonida and the Coding of Biological Information for Numerical Taxonomy." *Systematic Zoology*, Vol. 13, pp. 32–41.
HODSON, F. R., SNEATH, P. H. A. and DORAN, J. E. A study of bronze fibulae from the La Tène culture (to be published).
ROSE, M. J. (1964). Classification of a Set of Elements, *Computer Journal*, Vol. 7, pp. 208–211.
SOKAL, R. R. and SNEATH, P. H. A. (1963). *Principles of Numerical Taxonomy.* (San Francisco and London: W. H. Freeman, xvi + 359 pp. illust.).

Correspondence

To the Editor,
The Computer Journal.

Nonlinear programming test problems

Sir,

In a recent article by M. J. Box* on constrained optimization there is included a small five-variable nonlinear programming problem which is described as a very difficult problem. Since it is one of the few examples of a nonlinear programming problem given in the literature, it is likely to be used as a test problem by other persons working in the area of nonlinear programming. Persons using it as a test problem ought to be aware that the problem is easily converted to a linear programming problem. The form of this problem given in Box's article obscures the fact that this can be done by a simple transformation. It was in a discussion of this problem with Professor C. E. Lemke, of Rensselaer Polytechnic Institute, that we noticed this simple transformation.

The problem can be rewritten as follows:

$$\min_x \left\{ f(x) = b_0 + a_{01}x_1 + \left(\sum_{j=2}^{5} a_{0j}x_j\right)x_1 \right\}$$

subject to

$$0 \leqslant a_{i1}x_1 + \left(\sum_{j=2}^{5} a_{ij}x_j\right)x_1 \leqslant b_i \quad i = 1, 2, 3$$

$$x_1 \geqslant 0,\ 1{\cdot}2 \leqslant x_2 \leqslant 2{\cdot}4,\ 20{\cdot}0 \leqslant x_3 \leqslant 60$$

$$9{\cdot}0 \leqslant x_4 \leqslant 9{\cdot}3,\ 6{\cdot}5 \leqslant x_5 \leqslant 7{\cdot}0.$$

Then by letting $y_i = x_1 x_i$, $i = 2, 3, 4, 5$ and $y_1 = x_1$ we obtain the following linear programming problem:

$$\min_y \left\{ g(y) = b_0 + \sum_{j=1}^{5} a_{0j}y_j \right\}$$

$$0 \leqslant \sum_{j=1}^{5} a_{ij}y_j \leqslant b_i \quad i = 1, 2, 3$$

$$y_i \geqslant 0 \; i = 1, 5$$

$$y_2 - 1{\cdot}2\,y_1 \geqslant 0, \qquad 2{\cdot}4\,y_1 - y_2 \geqslant 0$$

$$y_3 - 20{\cdot}0\,y_1 \geqslant 0, \qquad 60{\cdot}0\,y_1 - y_3 \geqslant 0$$

$$y_4 - 9{\cdot}0\,y_1 \geqslant 0, \qquad 9{\cdot}3\,y_1 - y_4 \geqslant 0$$

$$y_5 - 6{\cdot}5\,y_1 \geqslant 0, \qquad 7{\cdot}0\,y_1 - y_5 \geqslant 0.$$

I solved this problem on the IBM 7040 using linear programming routine LP-40, and obtained the following optimal solution after six simplex iterations:

$$g = -5{,}280{,}344{\cdot}9$$
$$y_1 = 4{\cdot}53743,\ y_2 = 10{\cdot}88983,\ y_3 = 272{\cdot}24584$$
$$y_4 = 42{\cdot}19811,\ y_5 = 31{\cdot}76202$$

which in terms of the original variables gives $f = -5{,}280{,}344{\cdot}9$

$$x_1 = 4{\cdot}53743,\ x_2 = 2{\cdot}40000,\ x_3 = 60{\cdot}00000$$
$$x_4 = 9{\cdot}30000,\ x_5 = 7{\cdot}00000.$$

It is interesting to note that the original nonlinear problem is a non-convex problem; the feasible region is not a convex region nor is the objective function convex. Yet the transformation results in a convex, in this case linear, programming problem.

The values of the constants in this form are

$a_{01} = -8{,}720{,}288{\cdot}795$	$a_{21} = -155{,}011{\cdot}1055$
$a_{02} = -150{,}512{\cdot}524$	$a_{22} = 4{,}360{\cdot}5334$
$a_{03} = -156{\cdot}695$	$a_{23} = 12{\cdot}9492$
$a_{04} = -476{,}470{\cdot}319$	$a_{24} = 10{,}236{\cdot}8839$
$a_{05} = -729{,}482{\cdot}825$	$a_{25} = 13{,}176{\cdot}7859$
$a_{11} = -145{,}421{\cdot}4004$	$a_{31} = -326{,}669{\cdot}5059$
$a_{12} = 2{,}931{\cdot}1506$	$a_{32} = 7{,}390{\cdot}6840$
$a_{13} = -40{\cdot}4279$	$a_{33} = -27{\cdot}8987$
$a_{14} = 5{,}106{\cdot}1920$	$a_{34} = 16{,}643{\cdot}0759$
$a_{15} = 15{,}711{\cdot}3600$	$a_{35} = 30{,}988{\cdot}1459$
$b_0 = -24{,}345{\cdot}0$	$b_2 = 294{,}000{\cdot}0$
$b_1 = 294{,}000{\cdot}0$	$b_3 = 277{,}200{\cdot}0$

Sincerely yours,
W. CHARLES MYLANDER

Advanced Research Division,
Research Analysis Corp.,
McLean, Virginia, U.S.A.
3 August, 1965.

* Box, M. J. (1965). "A new method of constrained optimization and a comparison with other methods," *The Computer Journal*, Vol. 8, p. 42.

Downloaded from https://academic.oup.com/comjnl/article/8/4/383/400875 by guest on 13 March 2024