DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
University of Georgia	Unclassified
	2b. GROUP Unclassified

3. REPORT TITLE

"Configuration and Classification of Clusters in n-Dimensions"

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

Surendra J. Trivedi

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1971	156	38

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-69-A-0423	Technical Report No. 75
b. PROJECT NO. NR 042-261	Dept. of Statistics and Computer Science
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	No. 17 THEMIS

10. DISTRIBUTION STATEMENT

Reproduction in whole or part is permitted for any purpose of the United States Government.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	ONR - Washington, D.C.

13. ABSTRACT

Many experiments involve measuring a number of response variables simultaneously. As a result of giving one stimulus to an experimental unit, what we obtain is not just one response but several responses. In statistical language, we deal with a multivariate situation as opposed to univariate situations. Usually, many stimuli, called factors, are considered at many levels in the same experiment. Many statistical techniques are available to analyse this type of data and to draw conclusions therefrom. The present work considers one such technique, the identification of subgroups of individuals on the basis of responses, i.e., a special case of cluster analysis. Many different algorithms proposed for detecting clusters have been reviewed. These fall into two classes--(i) those which detect clusters of variables--factor analysis--and (ii) those which detect clusters of experimental units--cluster analysis. What we have done in the present work lies inbetween these two techniques. We first perform cluster analysis on p responses for each unit, and sort the experimental units into groups. After assigning experimental units to groups, we look for those individuals whose total response could be expressed in (p-1) combinations of original responses, irrespective of the groups or clusters to which they belong. Those individuals whose total response could be described in terms of (p-1) combinations of original responses are said to lie on a "simple structure plane." A geometric probability approach has been used to determine whether a given point lies on a simple structure plane. The orthogonal distance of a point from the plane is used as a criterion. If many points lie on a given simple structure plane, the plane is said to be overdetermined. Probability expression has been derived to determine whether a simple structure plane can be regarded as being overdetermined. Those individuals which lie on a (p-1)-dimensional hyperplane in a p-dimensional space, need one variable less in their description.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
	2b. GROUP

3. REPORT TITLE

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.		
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.		

10. DISTRIBUTION STATEMENT

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

Page 2.

In this case, (p-1) linear combinations of the original p variables, could be used for describing these points. For real life interpretation, this procedure would be useless as we would be left with (p-1) artificial variables instead of p observable variables. It would seem reasonable, then, to eliminate, for the data points close to one of the subspaces, that observable variable which contributes the least to the description of this selection of data points. A correlation approach has been used to determine the variable to be eliminated, and it turns out that the variable which correlates most strongly in absolute value with the artificial variable should be discarded.

Two computer programs have been developed to assist the user in analysing his data using this approach. The first one, named CLUSTR, identifies clusters and simple structure planes. The second one, an interactive graphics program, named ELLIPSE, can be used to visualize the configuration of clusters and the underlying simple structures. The clusters are projected onto many 2-dimensional spaces and displayed on the IBM 2250 Graphics terminal. If the plane of projection selected is orthogonal to a simple structure plane, the points lying on the particular simple structure plane, will make a band of narrow width more or less resembling a straight line. For other planes of projection this will not hold. The clustering of points is, however, unaffected by the choise of a particular plane of projection.

DD FORM 1 NOV 65 1473 (PAGE 1)

Unclassified
Security Classification

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
CLASSIFICATION						
CLUSTER ANALYSIS						
FACTOR ANALYSIS						
MULTIVARIATE ANALYSIS						
STATISTICAL COMPUTATION						
CONVERSATIONAL COMPUTATION						
GRAPHICS TERMINALS						

DD FORM 1 NOV 65 1473 (BACK)

Unclassified

THEMIS REPORT NUMBER 17

TECHNICAL REPORT NUMBER 75

CONFIGURATION AND CLASSIFICATION OF CLUSTERS IN n-DIMENSIONS

SURENDRA J. TRIVEDI

 This Research was supported, in part, by the Office of Naval Research, Contract Number N00014-69-A-0423, NR 042-261.

ROLF BARGMANN
PRINCIPAL INVESTIGATOR

THE UNIVERSITY OF GEORGIA
DEPARTMENT OF STATISTICS AND COMPUTER SCIENCE
ATHENS, GEORGIA 30601

DECEMBER 1971

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER I

INTRODUCTION

1.1 Multivariate Experiments

Many experiments involve measuring a number of response variables simultaneously. Thus, for example, in determining the effectiveness of a new drug, a person's systolic and diastolic blood pressure may be observed, before and after administration of the drug; also his pulse rate, temperature and other physiological data may be recorded. As a result of giving one stimulus to an experimental unit, what we obtain is not just one response but several responses. In statistical language, we deal with a multivariate situation (many responses) as opposed to univariate situations (only one response). An experiment is rarely so simple as described above. Usually, many stimuli, which will be called factors, are considered at many levels in the same experiment. Ultimately, the experimenter has a large collection of data before him. The problem is how to interpret the data. Depending upon the experimenter's objective, many statistical techniques are available to analyse the data and to draw conclusions therefrom. In the present work, we consider one such technique, the identification of subgroups of individuals on the basis of responses, i.e., a special case of cluster analysis. In the following sections, we review some of the problems of the subject and then outline the special problem in the present dissertation.

1.2 Review of the related literature

Consider an experiment where many responses are recorded on each experimental unit. We shall assume that there are n experimental units, and, on each unit, p responses are measured. The resulting data can be put in the form of an n x p matrix. Each row of this matrix corresponds to one experimental unit. Each experimental unit can be represented by a point in a p-dimensional space. It is possible that some of these points will be so close to one another that they form a "cluster." The problem of detecting clusters has been considered by many authors during the last 30 years. Tryon [37] in 1939 gave many algorithms based on the correlation matrix of variables, for the related problem of assigning variables to groups. The technique, much similar to the concept of the "coefficient of belonging" described by Harman [14], was developed on the assumption that correlations among variables belonging to the same group should be much higher than correlations between these variables and those not belonging to the group. Holzinger and Harman [15] defined their coefficient of belonging or B-coefficient as "100 times the ratio of the average of the intercorrelations among the variables of a group to their average correlation with all the remaining variables.

Sokal and Michener proposed the weighted mean-pair method to identify clusters. This method was originally applied to an entomological problem [33]. A more recent description of this method has been given by Sokal and Sneath [34] who recommend it as "the best of a class of commonly used methods of cluster analysis." The method operates on an N x N similarity matrix. Those two individuals, i and j say, which have the highest similarity are paired together, i.e., put in the same cluster.

Any appropriate measure of similarity e.g., product moment correlation coefficient, coefficient of association, etc., can be used, although by far the most commonly used measure is the product moment correlation coefficient. After the individuals i and j have been paired together, columns (and rows) i and j of the similarity matrix are replaced by a single column consisting of the means of the elements in rows (and columns) i and j. The process is then repeated on the new matrix of order (N-1), when either two new individuals have the highest similarity and form a new pair, or the existing pair combines with a further individual to make a cluster of three. The process continues and at each stage a new cluster consisting of a pair of individuals is formed or the new individual is assigned to one of the already existing clusters of previously combined individuals.

Edwards and Cavalli-Sforza [7] suggest dividing the points into two sets such that the sum of squares of distances between sets is a maximum. Thus according to this method, one can find only two clusters, no more and no less. Since the total sum of squares is a constant for a given sample, maximizing between-groups sum of squares is equivalent to minimizing the within-groups sum of squares. The method consists of examining all the $2^{N-1} - 1$ two-set partitions of N individuals and selecting the one which gives the minimum within-set sum of squares. The method is not suitable for a large value of N as the time required on a computer to examine all the two-set partitions is enormous. It was estimated that with N = 21, the time required to examine all the partitions, on a computer with 5 micro-second access time would be 100 hours and that for N = 41, it would be 54,000 years. Thus even with the help of the fastest computers, the method would be impracticable. One more algorithm

based on ecological applications was proposed by Williams and Lambert [38]. Gower [12] gives an excellent comparison of the last three mentioned algorithms together with some of his own modifications proposed for these algorithms.

Another important study was made by Neyman and Scott [27]. They extensively studied the clustering of galaxies in the universe and presented the theories of "simple clustering" and "multiple clustering." "Simple clustering" was based on the assumption that galaxies occur in clusters and that the cluster centers are uniformly distributed throughout the universe. In "multiple clustering" it was assumed that the cluster centers radiate from super clusters.

From the above discussion, it is clear that in the statistical literature, we come across two types of clusters--(i) the clusters of variables and (ii) the clusters of individuals or points or experimental units. Without going into the details of the confusion that these two concepts have created and their uses and misuses, we only note that given a multivariate sample, it is possible to identify the underlying clusters by applying any of the suitable techniques available.

We now describe in detail an algorithm proposed by Bargmann and Graney [5] to determine clusters with the object of identifying mixed samples of multivariate normal distributions. With the help of methods to be developed in the present work, we may then study the configuration of such subgroups.

1.3 Real and Virtual Clusters

Real clusters are defined to be clusters of points in the original space. Virtual clusters on the other hand are clusters of projections of

points in a space of dimension lower than that of the original space. To borrow an example from Graney [10], if one were to look for clusters of stars as observed from the earth, one would be dealing with virtual clustering. The observer perceives the stars as projections onto the surface of the celestial sphere. To determine the real clusters of stars, it would be necessary to measure the distance of each star from the observer. It is also obvious from the above example that, virtual clusters may not necessarily be real clusters, and vice versa. One may tend to think that real clusters would necessarily be virtual clusters also. This, however, is not so. If one were standing in the midst of a real cluster, he may not find any cluster at all. In this connection, it should also be noted that in the algorithm proposed by Graney [10], if a cluster is centered at the center of the entire system, it would not be detected. This is similar to identification of galaxies. Being a member of our own galaxy, we do not obtain, by direct observation, a description of the configuration of our galaxy. For that purpose, we will have to make calculations based upon distance measurements (or observe from a different galaxy).

1.4 d-clusters and k-clusters

As experiemental units are assigned to clusters, a decision has to be made whether two units are close enough to justify their inclusion in the same cluster. One may look at this problem in two directions: The so-called d-clusters and k-clusters. Consider a region S of fixed radius d. This is said to be overdetermined at the α-level of significance if the number of points in the region exceeds a value k_o such that, under the null hypothesis of uniform distribution of points,

$$P\ [k_o \text{ or more points in } S] < \alpha$$

This type of cluster is known as d-cluster since it results from a region of fixed radius d.

The other type of clustering results when a fixed number of points fall into a region with sufficiently small radius. Let there be a fixed number of points, say, k. Let d_o be the radius of a sphere (or hyper-sphere) just sufficient to enclose all these k points within the sphere. It is apparent that d, the radius of the sphere is a random variable. If

$$P\ [d < d_o] < \alpha$$

where α is the predetermined level of significance, then these k points are said to form an overdetermined cluster. Such a cluster is known as the k-cluster since it results from a fixed number of points falling within a sufficiently small sphere.

1.5 Identification of Mixed Samples

Consider the case of k-cluster discussed above. As an example of this type of cluster, consider an experiment in which the number of people given a drug from a certain class of drugs (which produce similar effects) is limited. Then one would like to know if the symptoms among some individuals are more closely alike than those of other individuals. In other words, it would be appropriate to see if there are some individuals whose symptoms are so close that they form a cluster. But this also implies that we are looking for a principle of classification which distinguishes this group of individuals. Such a situation can arise in linear analysis. For the sake of simplicity, we shall describe the situation in terms of univariate analysis, but it applies equally well to multivariate analysis. In a two-factor experiment one being applied at r levels and the other being

applied at s levels, we will have r x s cells; let us assume that there are n_{ij} observations in cell (i, j). Analysis of these data on the basis of a linear statistical model assumes homogeneity of cell variances. The hypothesis of the equality of cell variances can be tested by application of Bartlett's test. If this hypothesis should be rejected, three possible causes can be responsible: (i) The cell variances are not constant, σ^2, but proportional to some known v_{ij} (a different one for every cell). A variance-stabilization transformation, or weighted regression, or both, can be employed to correct this situation. (ii) There may be "mavericks" --misclassified or improperly recorded items. They can be omitted before the analysis of the data. (iii) There may be a third factor of classification present. If this is the case, what we regard as "error sum of squares" is in fact not the error sum of squares but the variance component sum of squares error, plus sum of squares due to the third factor which we have not taken into account. In multivariate analysis of variance, this quantity would be the (H + E) matrix instead of the E matrix alone where E stands for the "error" SSP matrix and H stands for the "hypothesis" SSP matrix.[1] Hence the problem reduces to that of detecting the third hidden factor. It is clear that the sample within each cell comes from two or more populations instead of from just one. It will be necessary to "unmix" the samples within each cell before a valid linear statistical analysis of the data can be performed. Instead of describing the technique of unmixing the mixed samples, we refer to Graney [10], for a full description of the technique, which also contains a number of illustrations.

---

[1] SSP - sum of squares and products, sometimes also called "Wishart" matrix.

There has been, in recent years, a considerable resurgence of interest in cluster analysis. Ling [23] has discussed many techniques and he has tried to classify these. It is apparent that there is little purpose in inventing yet new similarity indices, distance measures, or search algorithms. The present dissertation deals with a point intermediate to the two problems which cluster analysis has attempted: (a) classification of individuals, on the basis of responses and (b) classification of response variables assuming a homogeneous group of individuals (really factor analysis). Cattell [6], and Stephenson [35] view the entire complex as "factor analysis" and call the first problem the "Q technique," the second problem the "R technique," and the combined problem, the "P technique." Unfortunately, their techniques do not lend themselves to a study of configurations because in their attempt to regard every problem as a correlational one, the authors perform analyses which become self-contradictory. For example, sums of squares and products can be used as terms in estimates of correlation between variables only if the individuals are <u>independent</u>, and conversely, "correlations" between individuals can be estimated by a product-moment approach only if the variables are uncorrelated. Quadratic forms would be needed otherwise, and the sums of squares and products can be quite meaningless. The present study avoids this confusion by separating, at each stage, the clustering problem (cluster of individuals) from the problem of structures of <u>variables</u> within clusters.

Guttman [11] has proposed a yet another technique to reduce the dimensionality of data. The technique, "nonmetric" in nature, works on an $n \times n$ symmetric matrix R, and determines those transformations which yield

an Euclidean coordinate system X (X : n x m), such that $XX' = F$ for m a minimum and, furthermore, satisfy all inequalities that whenever $r_{ij} \gtrless r_{kl}$ then $f_{ij} \gtrless f_{kl}$ for the non-diagonal elements of R and F, $(i \neq j, k \neq l)$. This avoids the problem of communalities and "when some lawful structure or pattern is present in the data, e.g., a simplex, a circumplex, or a radex, a nonmetric analysis will reveal the configuration, whereas a metric approach will obscure the lawfulness." For detailed discussion of this approach and the algorithms developed in this connection, refer to Guttman [11] and Lingoes and Guttman [24]. It is clear that this technique will reduce the dimensionality of all the data points. Again, as in factor analysis, it will not be affected by mean shifts. It is thus an alternative to structural and factor analyses and not an "in-between" solution as proposed by us in section 1.6.

1.6 Definition of the Problem:

In the above sections, we have given a brief outline of traditional approaches to cluster analysis. The discussion reveals that, given a multivariate sample, it is always possible to detect some underlying cluster structure and to assign points (experimental units) to the clusters to which they belong. This is not the only kind of analysis that can be performed. The same data could also be subjected to factor analysis, which assigns response variables to classes. Cluster analysis will group the individuals bringing out the mean effects and leaving the variable structure unaltered. Factor analysis will describe the data in terms of artificial variables without telling anything about the group means involved. What we propose to do in the present work lies in-between these two extremes. We first perform cluster analysis on p responses for

each unit, and sort the experimental units into groups. After assigning experimental units into groups, we look for those individuals whose total response could be expressed in (p-1) combinations of original responses, irrespective of the groups or clusters to which they belong. Thus the ultimate purpose of our analysis is to elicit more information from a set of multivariate data than is possible by cluster or factor analysis alone. Frequently both of these extremes produce trivial results. In medical applications, cluster analysis tends to producing clusters of patients who are healthy, slightly ill, and very ill. By contrast, factor analysis identifies a collective set of symptoms such as "fever," "pain," and "chills."

The algorithm developed in the present work begins by identifying clusters in the sense of estimating parameters of mixed multivariate normal distributions with equal variance-covariance matrices. If this were not done the unit ellipsoids around the grand mean would be affected, uncontrollably, by mean shifts. Within each of these ellipsoids, we look for well overdetermined subspaces of lower dimensionality, in the sense that we want a very significant number of individuals to fall into a region close to these highly overdetermined subspaces. These will be called "Simple Structure Planes."

We use this term because of its similarity with one criterion which Thurstone [36] used in describing a "simple structure" in the common-factor space of factor analysis. If all principal components of the within-cluster matrix were calculated and "rotated," the results of our study could also be produced. This, however, would be a tremendously wasteful procedure.

Those individuals which have a large distance from the subspace, yet belong to the original cluster, would differ from the others in that they require at least one additional diagnostic.

Geometrically, the Euclidean distance on a metric given by the unit ellipsoid, is obtained as follows: A vector is passed from the center of the ellipsoid (O) to the point in question (A). This vector intercepts the unit ellipsoid at point (S). The Euclidean distance is then (length OA)/(length OS). For this reason, the ellipsoid is called "unit ellipsoid." It generalizes the concept of a "unit interval" in one dimension (hence metric). The computer output reports these distances as a vector $\underline{v}$.

Now, the largest projected distance of a point, onto a 2-dimensional subspace, from the simple structure unit ellipsoid, is at most equal to the actual distance in p dimensions. Hence a point showing an appreciable distance from the simple structure ellipsoid, on any projection, would be representative of a point requiring one variable more, for adequate description, than the points lying in the simple structure region. This property is not related to the clustering of the points. Points in the same simple structure subspace may be far removed from each other; they may belong to different clusters.

The points having a large (positive or negative) distance from each of these planes can now be scrutinized. They share some characteristics, which makes them different from the other points on this plane. It is to be noted here that this procedure was not intended to find subclusters--one could do that by tightening the control constants in the

original program--but to study subspaces of lower dimensionality.[2] Thus the multivariate data are viewed from a new point of reference, and one may identify principles permitting a different taxonomy of experimental units (patients, plants, etc.).

There is a certain analogy of techniques between the subspace solution and the "simple structure rotation" in factor analysis. This is expected in virtual clusters. The cosine of the angular distance between two vectors can be regarded as a correlation if the vectors represent variables. It is in this sense that our n data points correspond to n correlated variables of factor analysis and our variables themselves correspond to the factors. With this understanding we can apply the rotational techniques to the original data matrix in order to obtain points lying on simple structure planes.

In the present dissertation, we have synthesized these two techniques--cluster analysis and simple structure identification--into a single program. When the user subjects his data to this program, named CLUSTR, he gets clusters and simple structure planes as the output. Next, we have developed an interactive graphics program, named ELLIPSE, which can be used to visualize the configuration of clusters and the underlying simple structures. Configuration of multidimensional clusters can be determined only if their dimensionality is reduced. For this purpose, it is necessary to project the clusters onto several 2 or 3 dimensional subspaces. The emphasis in the present work is on projecting the clusters

---

[2] Those readers who assume that this is analogous to factor analysis should be reminded that the latter increases the dimensionality from p correlated to p + k (at least partially) uncorrelated variables. There is, of course, no relationship to an incomplete "component analysis" which produces singular solutions.

onto many 2-dimensional spaces and displaying them on the IBM 2250 Graphics terminal. If the plane of projection selected is orthogonal to a simple structure plane, the points lying on the particular simple structure plane will make a band of narrow width more or less resembling a straight line. The display program can also be used in an exploratory manner. The user can supply many different vectors. If, by using some vector of projection, he sees narrow bands as described above, the corresponding vector is orthogonal to a simple structure plane. Such visual determination of simple structure planes, however, is rather difficult especially if the user is required to make inferences on configurations in four or more dimensions, on the basis of 2-dimensional displays. It is implicit from the above discussion that the determination of simple structure is equivalent to the fact that the points lying on a simple structure plane need at least one variable less in their description. Thus if p variables have been measured on all the experimental units of the sample (p-1) variables are adequate in the case of those experimental units which lie on a simple structure plane. In the case of these experimental units, one of the p variables can then be expressed by a linear combination of the remaining (p-1) variables. The statistical interpretation of this phenomenon is considered in Chapter 3.

CHAPTER II

PRINCIPAL COMPONENT AND OTHER PROJECTIONS

2.1 Introduction

As stated in the previous chapter, one of the goals of the present work is to display many different projections of multidimensional clusters. In this chapter, we give an account of projections along eigenvectors or principal components of the metric ellipsoids onto 2-dimensional subspaces. The principal component projections are necessarily "orthogonal." It is worth noting at the outset that very little information was gained by these principal component projections. Not that we were surprised by this finding, but some social scientists seem to attribute a lot more to this particular mathematical reference frame than it deserves.

2.2 Principal Component Projections

Let $Z = (Z : n \times p)$ be a data matrix of the multivariate observations. The n data points can be represented in a p-dimensional space. It is assumed that the clusters formed by these n points have been identified as well as the points belonging to the clusters. It is further assumed that points, which cannot be assigned to one of these clusters, have been eliminated. Now, if the data come from a p-variate normal distribution (or a mixture of p-variate normal distributions with different mean vectors but the same variance-covariance matrix), the clusters would be elliptical in shape, and the ellipsoids would have the same orientation. If there are k clusters, and if we denote by $\underline{\mu}_1, \underline{\mu}_2, \ldots, \underline{\mu}_k$ the

cluster centers, the equations of the ellipsoids enclosing these clusters can be written as

$$(\underline{x} - \underline{\mu}_i)' \Sigma^{-1} (\underline{x} - \underline{\mu}_i) = \text{constant}$$

for $i = 1, 2, \ldots, k$, where $\underline{x}$ stands for the running coordinates. The points belonging to a particular cluster will be enclosed within the respective ellipsoids. In practice, since the population (or populations) from which the sample was drawn, will not be exactly normal, we will not expect all the data points belonging to a particular cluster to lie within the corresponding ellipsoid. However, unless the population is far from normal, the ellipsoidal fit will be quite good. To visualize how the points are distributed in p-dimensional space and how they look in relation to their enclosing ellipsoids, we need to project the points onto several 2-dimensional spaces. The technique is simple and classic and is spelled out here for the sake of completeness. Let us take the equation of the ith unit ellipsoid[1]

$$(\underline{x} - \underline{\mu}_i)' \Sigma^{-1} (\underline{x} - \underline{\mu}_i) = 1 \qquad (2.2.1)$$

Since Σ is a positive definite (or at least positive semi-definite) matrix, there exists an orthogonal matrix Q such that

$$Q' \Sigma Q = D_\lambda \qquad (2.2.2)$$

where D_λ is a diagonal matrix with diagonal elements equal to the characteristic roots of Σ, and Q is the matrix of the eigenvectors of Σ. (2.2.2)

---

[1]This generalizes the "standard unit interval" in univariate analysis.

can be written as

$$\Sigma = Q D_{\lambda} Q' \qquad (2.2.3)$$

and hence

$$\Sigma^{-1} = Q D_{1/\lambda} Q \qquad (2.2.4)$$

where $D_{1/\lambda}$ is the inverse of D_{λ}. (The reciprocals of the characteristic roots of Σ are the diagonal elements of $D_{1/\lambda}$). Substituting (2.2.4) into (2.2.1), we have

$$(\underline{x} - \underline{\mu}_i)' Q D_{1/\lambda} Q' (\underline{x} - \underline{\mu}_i) = 1 \qquad (2.2.5)$$

or, if we let $\underline{y}'_i = (\underline{x} - \underline{\mu}_i)' Q$, this reduces to

$$\underline{y}'_i D_{1/\lambda} \underline{y}_i = 1 \qquad (2.2.6)$$

(2.2.6) is the standard principal axes reduction of the conic (2.2.1). The transformation $\underline{y}'_i = (\underline{x} - \underline{\mu}_i)' Q$ is the orthogonal rotation of the original reference axes in the direction of the principal axes of the ellipsoids. The direction cosines of the principal axes of all the k ellipsoids are identical because of the assumption of equal metric (homogeneity of dispersion matrices). If we write the matrix Q as

$$Q = [\underline{q}_1, \underline{q}_2, \ldots \underline{q}_p]$$

where $\underline{q}_1, \underline{q}_2, \ldots, \underline{q}_p$ are the eigenvectors of Σ, the transformation $\underline{y}'_i = (\underline{x} - \underline{\mu}_i)' Q$ can be written as

$$\underline{y}'_i = (\underline{x} - \underline{\mu}_i)' [\underline{q}_1, \underline{q}_2, \ldots, \underline{q}_p].$$

Since Q is orthogonal, $\underline{z}'_i Q$ will be coordinates of an original data point $\underline{z}'_i$, which is the ith row of the data matrix Z, with reference to the new coordinate axes. In particular $\underline{z}'_i \underline{q}_1$, $\underline{z}'_i \underline{q}_2$ will represent the

orthogonal projection of the original data point $\underline{z}'_i$ onto the 2-dimensional plane determined by the eigenvectors $\underline{q}_1$ and $\underline{q}_2$. Let us assume that the eigenvectors are arranged in descending order, i.e., $\underline{q}_1$ is the eigenvector corresponding to the largest root, $\underline{q}_2$ that corresponding to the second largest root, etc. Then the 2-dimensional plane determined by $\underline{q}_1$ and $\underline{q}_2$ is the plane containing the two largest principal axes of the ellipsoids and we would be projecting the data points onto this plane. We can take all the $\binom{p}{2} = p(p-1)/2$ pairs of eigenvectors and project the data points on these planes.

2.3 Principal Component Displays

The IBM 2250 Graphics terminal was used to display projections on the $p(p-1)/2$ 2-dimensional planes formed by each pair of the eigenvectors. The data used by Graney [10] were taken, and the three clusters identified by him were projected on all the three 2-dimensional pairs formed by the 3 variables. The unit ellipses (i.e., projections of the unit ellipsoid) around the clusters were also displayed. The orientation of these ellipses, is, in the case of principal components, of course parallel to the axes of reference. If, however, one of the axes is not a principal component, the inclination of the principal axes of the ellipses with the reference axes can be displayed. This inclination may present some evidence regarding the nature of the data points, which was obscured in the principal component plot. Because of the disappointing lack of information contained in the principal component plots, we did not even bother to document the computer program. Instead, the program which has been documented permits projection around arbitrary pairs of reference axes. These computer programs are described in detail in Chapter V.

2.4 Other Projections

Let the equation of the n-dimensional unit ellipsoid be

$$(\underline{x} - \underline{\mu}_i)' \Sigma^{-1} (\underline{x} - \underline{\mu}_i) = 1$$

where Σ is the p x p variance-covariance matrix, $\underline{x}$ are the running coordinates and $\underline{\mu}_i$ is the cluster center of the ith cluster. There will be as many ellipsoids as the number of clusters identified. For the sake of notational convenience, we will drop the subscript i from $\underline{\mu}_i$. As all the ellipsoids are referred to the same metric Σ^{-1}, they differ only with respect to their location. In the discussion that follows, we are concerned with the shape rather than the location. The matrix Σ, as a population parameter, is unknown and we will replace it by its unbiased estimate, s, the matrix of mean squares and mean products within groups. The projection on the 2-dimensional plane formed by the variables i and j can be written as

$$s^{ii}(x_i - \mu_i)^2 + s^{jj}(x_j - \mu_j)^2 + 2\, s^{ij}(x_i - \mu_i)(x_j - \mu_j) = 1$$

where s^{ij} is the (i, j)th element of s^{-1}. These equations for various values of i and j are employed in displaying the projections. For the purposes of computer programming, this equation is further simplified as

$$s^{ii}y_i^2 + s^{jj}y_j^2 + 2s^{ij}y_iy_j = 1$$

where $x_i - \mu_i = y_i$ and $x_j - \mu_j = y_j$. By employing the standard reduction techniques, the coordinates to plot the ellipses can be easily calculated. Further aspects on this phase of the computer program are treated in Chapter V.

CHAPTER III

CLUSTERS IN SUBSPACES--THE SIMPLE STRUCTURE SUBSPACE

3.1 The Problem

The previous two chapters considered the problem of cluster identification and cluster configuration. We now come to the second part of our inquiry--identification of experimental units or points which could be described by measuring a fewer number of variables on them. Before we proceed further with the identification of the points lying on a subspace, we want to consider the situations where such a problem can arise. A familiar example would be one of medical diagnosis. A number of patients are measured on a number of medical symptoms. Sometimes these measurements are repeated on the same patients for a number of days consecutively. Here the symptoms measured are our variables and the patients are subjects or experimental units. If the measurements are taken on different days, it would add a factor of classification; let us assume that this factor (i.e., days) has not been recorded. Of course, from these data one can construct a variance-covariance matrix and from that obtain a correlation matrix. This could be subjected to factor analysis. Factor analysis would reveal groupings of the symptoms in these data. Application of Thurstone's [36] principle of simple structure could reveal such groupings. Disappointing examples of this kind of analysis resulting in the symptoms like "chill," "fever," and "pain" are not so uncommon.

The data could also be subjected to cluster analysis to form groups of patients. In this type of analysis, the patients would be classified into groups depending upon the absence or severity of symptoms they have in common with respect to other patients belonging to the same group. Thus, for the patients belonging to the same group, almost all the patients would be measuring equally on the average on different symptoms; they may either measure high on the same symptom compared to other patients belonging to different groups or measure low, etc. To summarize, factor analysis would tell us about the symptoms, and cluster analysis would help us to group patients according to the "degree of severity," say, of the symptoms. According to cluster analysis, we may have two patients belonging to different groups perhaps because one measured low on one symptom and the other measured high on the same symptom. It may happen that the particular symptom would have left the final diagnosis unchanged. To this extent, this symptom could be discarded so far as these two individuals are concerned and then they will belong to the same "group." But neither the factor analysis nor cluster analysis would bring out this fact; nor would it bring out the fact that "days" is a hidden factor. We must extend both techniques before we can identify such configurations. The problem is formulated in mathematical terms in the next section where we present, in detail, a specific approach.

3.2 Mathematical Formulation

In previous chapters we have dealt with the techniques of cluster analysis. It was pointed out that given an $n \times p$ data matrix, it is possible to identify the underlying clusters. We now ask the next

question--are there any data points which instead of lying in the p-dimensional space, lie on the (p-1) dimensional hyperplane? This question is important as an answer to it, among other things, will reveal the following things: (i) The points that lie on the (p-1)-dimensional hyperplane. The determination of the points lying on the (p-1)-dimensional hyperplane will help us to describe these points in terms of (p-1) variables instead of p variables. (ii) We will essentially devise a "discriminant function" which helps us to split the original sample into two groups of points--one group which needs all the p original variables to describe the points belonging to it and another group which needs (p-1) instead of p variables to describe the points belonging to it. In the second case, we have also to consider the question of which variable to discard from the original p variables. Note that it is also implied in the second case that the p x p variance-covariance matrix of the points lying on the (p-1)-dimensional hyperplane will be singular, as its rank will be (p-1) and not p.

3.3 A Basic Probability

At the outset, let us make clear what we mean by points lying "approximately" in a subspace. Our attempt will be to look for those points which lie close to a (p-1)-dimensional hyperplane in a p-dimensional space. Clearly, any (p-1) vectors always lie exactly on a (p-1)-dimensional hyperplane, whereas a minimum of p vectors is necessary to overdetermine a (p-1)-dimensional hyperplane. We will, therefore, look for those (p-1)-dimensional hyperplanes which have p or more vectors lying close to them. The singular case where p or more vectors lie, exactly, on a plane will be ignored, since in this instance, those points lying on the hyperplane will

satisfy a linear relationship between the p-variables; this cannot happen unless there is a deterministic linear relationship between the p-variables in the population, and thus, any sample will reflect this relationship and the $p \times p$ estimated variance-covariance matrix of any sample from such a population will be singular. Since we require inverses of dispersion matrices, we must exclude these redundancies. To determine, whether a point overdetermines a hyperplane, we shall consider its Euclidean distance from the hyperplane and examine whether this distance could be considered negligible in a probabilistic sense. We need an expression for this probability. The derivation follows the reasoning given by Bargmann [2]. Let P be a point in 2-dimensional space. We are interested in the orthogonal distance of this point from a line, which is a hyperplane in 2-dimensional case. Without loss of generality, we can assume the X-axis to be this line (Figure 3.3.1). Let the vector OP subtend an angle θ at the origin with the X-axis. We must now assume that the reference axes span a Cartesian frame (orthogonal, equal units along each axis). This implies that a Gram-Schmidt transformation of the original observations (using S, the within estimated dispersion matrix as the original metric) must precede the calculation of probabilities of overdetermination. With this assumption we can now draw a circle with OP as radius. Let M be the foot of the perpendicular drawn from P on the X-axis. Then

$$PM = OP \cdot \sin \theta$$

Therefore, $\theta = \sin^{-1} PM/OP = \sin^{-1} 2PM/2OP = \sin^{-1} a/2h$

Where $a = PP'$ and h is the radius of the circle. Thus, the probability that a point falls within the angle θ on the circumference of the circle is given by

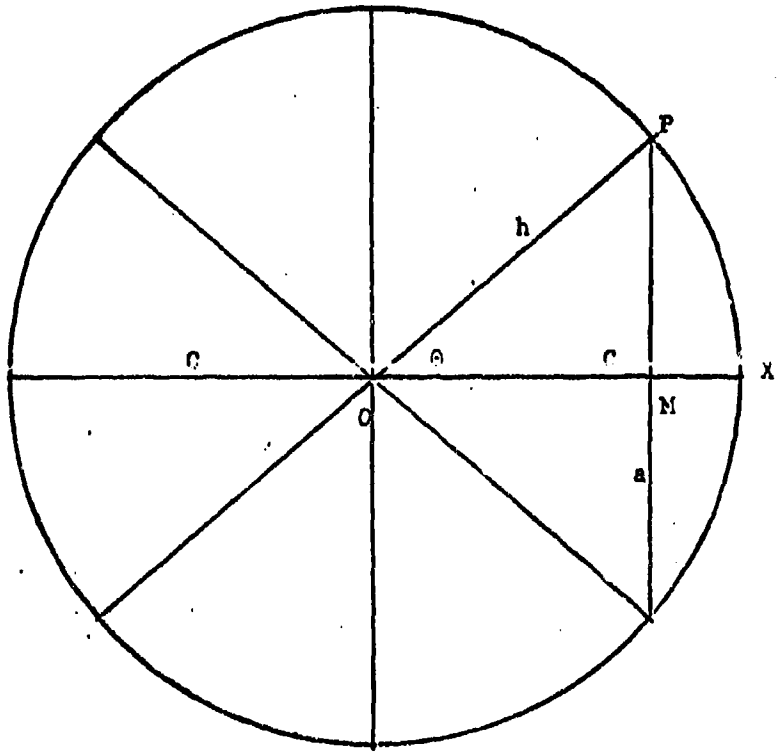


Figure 3.3.1

$$P_2 = \frac{2 \times \text{length of arc generated by angle } 2\theta}{\text{circumference of the circle}}$$

$$= 2 \cdot OP \cdot 2\theta / 2\Pi \cdot OP$$

$$= 2\theta/\Pi$$

$$= (2/\Pi)\arcsin(a/2h) \qquad (3.3.1)$$

We can view the above probability either way--

(i) the probability of a point falling within an angle θ as stated above

(ii) the probability of a point falling within the orthogonal distance of $+a/2$ to $-a/2$.

The technique involved in generalizing the above probability to higher dimensions, say k, is simple. We have to obtain the surface element of a sphere of radius h in k dimensions and the surface element cut off by the k-dimensional arc which subtends an angle θ at the center of the sphere. We shall illustrate the procedure for 3-dimensions before generalizing it to k-dimensions. The 3-dimensional sphere can be obtained by revolving a semicircle around its diameter. The surface element generated by arc between $y = -a/2$ and $y = a/2$ is twice the element generated by arc between $y = 0$ and $y = a/2$ and hence the 3-dimensional surface element between $y = -a/2$ and $y = +a/2$ can be expressed as

$$C_3 = 2 \cdot 2\Pi \int_0^{a/2} x\,ds$$

where $x = \sqrt{h^2 - y^2}$ and $ds = h\,dy/\sqrt{h^2 - y^2}$

Hence $C_3 = 2 \cdot 2\Pi \int_0^{a/2} (h\sqrt{h^2 - y^2})/(\sqrt{h^2 - y^2})\,dy$

$$= 2\Pi a h$$

and S_3, the total surface element will be

$$2 \cdot 2\pi \int_0^h (h \cdot \sqrt{h^2 - y^2}) / (\sqrt{h^2 - y^2})\, dy$$

$$= 4\pi h^2$$

Hence, $P_3 = C_3/S_3 = a/2h$ (3.3.2)

For k dimensions, the (k-1)-dimensional semisphere is required to be revolved around a (k-2)-dimensional hyperplane. In the above notation, the probability P_k can be expressed as

$$P_k = \frac{2 \cdot \frac{S_{k-1}}{h^{k-2}} \cdot h \cdot \int_0^{a/2} \frac{(\sqrt{h^2 - y^2})^{k-2}}{\sqrt{h^2 - y^2}}\, dy}{2 \cdot \frac{S_{k-1}}{h^{k-2}} \cdot h \cdot \int_0^{h} \frac{(\sqrt{h^2 - y^2})^{k-2}}{\sqrt{h^2 - y^2}}\, dy}$$

$$= \frac{\int_0^{a/2} (\sqrt{h^2 - y^2})^{k-3}\, dy}{\int_0^{h} (\sqrt{h^2 - y^2})^{k-3}\, dy}$$

In the numerator of the above expression, substitute $y = h\sin\phi$. Then, $dy = h\cos\phi d\phi$ and the integral reduces to

$$h^{k-2} \int_0^{\alpha/2} \cos^{k-2}\phi d\phi$$

where $\alpha/2 = \arcsin a/2h$. By the same substitution, the denominator could be reduced to

$$h^{k-2}\int_0^{\Pi/2} \cos^{k-2}\phi d\phi$$

If we let $\sin^2\phi = z$, this integral can be reduced to the complete Beta integral

$$\frac{1}{2}B\left(\frac{1}{2}, \frac{k-1}{2}\right) = \frac{1}{2}\int_0^1 z^{-1/2}\ (1-z)^{(k-3)/2} dz$$

and the denominator could be reduced to

$$\frac{1}{2}\int_0^{\sin^2\alpha/2} z^{-1/2}\ (1-z)^{(k-3)/2}\ dz$$

Therefore, P_k, the ratio can be expressed as

$$P_k = B(\sin^2\alpha/2;\ 1/2,\ (k-1)/2) \qquad (3.3.3)$$

where the right side of (3.3.3) is the incomplete Beta function. This is the probability of a point falling within (-a/2, +a/2) in k dimensions. We will now make use of this probability expression in studying the simple structure configuration.

3.4 Determination of Simple Structure Configuration

Let us assume that we have a total of n points in p-dimensional space. As stated in the previous section, the probability, in p-dimensional space, of a point falling within (-a/2, a/2) is given by

$$P_p = B(\sin^2\alpha/2;\ \frac{1}{2},\ (p-1)/2) \qquad (3.4.1)$$

where $\alpha/2 = \arcsin a/2h$. Without loss of generality, h could be taken to be 1. We are interested in determining whether these are points lying on a

subspace. In a p-dimensional space, (p-1) points always lie on a (p-1)-dimensional subspace. Thus out of a total of n points, we have available only (n-p+1) free points. Likewise, if we are to make any statement about r points lying on a subspace, we can only look to (r-p+1) points as (p-1) of them will always lie on a subspace. If we denote by P_p the probability of a single point lying within a fixed distance $\pm a/2$ from a given (p-1)-dimensional hypersphere, in p-space, the probability that (r-p+1) points will fall in that region out of (n-p+1) is given by

$$\binom{n-p+1}{r-p+1} p^{r-p+1}\, 1-p_p^{n-r}$$

Thus, if a region is to contain more than r points, the probability would be given by the cumulative binomial distribution

$$\sum_{i=r}^{n} \binom{n-p+1}{r-p+1} p_p^{i-p+1} (1-P_p)^{n-i} \qquad (3.4.2)$$

Also on the basis of the (n-p+1) free points available, the expected number of points (in excess of (p-1) falling in a (p-1)-dimensional region is

$$(n-p+1)P_p \qquad (3.4.3)$$

The sum of binomial terms (3.4.2) can be evaluated by the incomplete Beta function. The result can be summarized as

$$P[\text{r or more of n points lie within } \pm a2] = B(P_p;\ r-p+1,\ n-r+1) \qquad (3.4.4)$$

where P_p itself is an incomplete Beta function. If this probability is very small we shall say that the simple structure subspace determined by

the points lying on it is overdetermined. However, the probability of a subspace being overdetermined itself depends on the probability P_p. We, therefore, need to consider the interval width and the probabilities, so that the "probability of overdetermination" may be a meaningful concept. Bargmann [2], in studying the overdetermined subspaces in relation to factor analysis fixed the ratio a/h to be ±0·10. These criteria, which reflect common usage in factor analysis, did not suit our requirements.

The object of this test, as will be discussed in Chapter IV, is to suggest to the viewer of a graphics display, some vectors which are normal to overdetermined hyperplanes. If no such concentration were present, the expected number of points, would be, (according to (3.4.3) and the discussion on free points)

$$(n-p+1)P_p + (p-1) \qquad (3.4.5)$$

After considerable experimentation, with n between 20 and 50, and p between 2 and 5, we found that taking P_p such that the expected number of points is (p+5), i.e.,

$$P_p = 6/(n-p+1) \qquad (3.4.6)$$

gave satisfactory results in the identification of overdetermined hyperplanes by the Beta test (3.4.4). A value of P_p smaller than this was too stringent so that a considerable number of points would have to lie on a subspace before it could be considered well determined. The suggested value of P_p was fairly moderate. We compensated for this value by tightening up the probability level for declaring a subspace to be overdetermined. We set this probability at 0.01.

The arbitrariness of choice of P_p and levels of significance may be disquieting to some readers. They may be reminded though, that we are dealing with a phase of data analysis which is exploratory. A sample from

a p-variate normal distribution, with a dispersion matrix of rank (p-1), will always be on a (p-1)-dimensional subspace, exactly. There is no test for a hypothesis in the population. Rather, we deal with an instance where, after some linear transformations of the original variables, one of them has negligible variance, after some points have been deleted from the sample. Consequently, a decision as to what is negligible, and what is "close to a plane" is really as arbitrary as declaring that, viewed from some point in the universe, most (but not all) of the stars of a galaxy lie close to a plane. In the final analysis, only the graphic display of certain projections will reveal such intuitive configurations.

In our examples, the subspaces were overdetermined at much smaller values than 0.01 which points to the fact that (3.4.6) was quite useful. For the rest, the choice of critical values is as arbitrary as "0.05 level" (because we have five fingers?) and the 0.01 level of significance. As a guide for displaying configurations, our two levels of P_p and α were "useful."

CHAPTER IV

CLUSTERS IN SUBSPACES--IDENTIFICATION

4.1 Introduction

In the previous chapter, we addressed ourselves to the statistical aspects of well determined subspaces and derived a few pertinent geometric probability expressions which may guide us to find such spaces. No mention was made of techniques for finding such configurations. In the present chapter, we consider (a) the technique of determining points lying on a subspace; and (b) the problem of which variable could be discarded for the points which describe an overdetermined subspace.

4.2 Overdetermined Subspaces

The problem of determining subspaces in our case is rather similar to the determination of simple structures in factor analysis. The essential difference, however, lies in the fact that a factor analyst looks for simple structure among variables. From data points, he constructs a correlation matrix and gets a factor matrix by applying one of the many suitable techniques available for this purpose. If he so desires, after obtaining an initial solution of the factor matrix, he may obtain a "preferred" representation, e.g., Lawley's form [14], etc. For a factor analyst such a solution may not serve his purpose if he is interested in relating the artificial variables to observable ones. In that case he will have to resort to some other forms of representation, such as the simple structure technique. The number of artificial variables required

to explain an observable variable is known as the complexity of the observable variable. For the purpose of interpretation of artificial variables, it is desirable that complexities of observable variables be low. Both analytic and geometrical techniques are available to a factor analyst to express the final solution in a form suitable for interpretation.

We have a slightly different problem. First of all, we do not look for any artificial variables to represent the observable variables. Thus whereas a factor analyst works on a factor matrix, we work on the data matrix itself. The starting point for a factor analyst is the correlation matrix obtained from the data matrix. A somewhat similar standardization is employed in our case, except that we standardize the data matrix on the basis of cluster means and the "within" matrix and then normalize the points to unit length. After displaying these points, and the unit ellipses around the cluster means, on all 2-dimensional directions, we proceed to single out those points which could be described in terms of fewer variables. In this connection, it does not concern us how far apart these points are, as long as they lie on a subspace of lower dimensionality. Thus this technique has an advantage that it can identify points lying on a subspace even though they may be belonging to different populations, or clusters. This is precisely what we had in mind. The technique of cluster analysis assigns points to the populations to which they belong. Leaving this structure intact, our new technique determines points which lie on a subspace.

A discussion may be in order regarding the number of subspaces one can find. If we can determine one overdetermined plane, the chances are that there are many planes in the vicinity of a plane already found.

The reason is that, by "tilting" the plane already found, we can still retain many of the points belonging to the original subspace found, pick up a few new points and obtain another overdetermined plane. However, we can reduce the multiplicity of subspaces by ignoring these additional planes found which lie within a certain range of the original plane. This can be done by requiring a minimum angle between the normals to two distinct planes. What angle should be maintained between two planes before declaring them as distinct is a matter of choice. The "orthogonality" preferred by some factor analysts is, at least for our problem, quite useless.

4.3 Identification of Subspaces

In this section, the general identification technique will be described. Let there be n experimental units, each with p measurements. The data matrix of order n x p will be designated as X. This matrix is subjected to a cluster analysis program. If the data are normally distributed, we need to use virtual clusters; hence we used the program developed by Bargmann and Graney [5] for this purpose. After NG (computer program notation) such clusters have been found, we may define a matrix A, with elements a_{ij}, of order n x NG such that

$$a_{ij} = 1 \quad \text{if unit } i \text{ belongs to cluster } j$$

$$= 0 \quad \text{otherwise} \qquad (4.3.1)$$

Let D_k (of order NG x NG) denote a diagonal matrix with elements k_j, $j = 1, 2, \ldots, NG$ where k_j is the number of points assigned to cluster j. Subtraction of cluster centers or cluster means from each point produces a matrix Y, formally given by,

$$Y = X - AD_k^{-1}A'X \tag{4.3.2}$$

$$= (I - AD_k^{-1}A')X \tag{4.3.3}$$

Now we can obtain an estimate of the common dispersion matrix Σ, using the within-cluster sample dispersion matrix

$$S = (1/n_e)Y'Y \tag{4.3.4}$$

where $n_e = n-NG$. For the determination of subspaces, we must first transform our reference frame to a Cartesian metric (which is, of course, merely a computational device, and never actually displayed). As a convenient technique, we used the Gram-Schmidt ("Forward Doolittle") reduction,

$$S = TT' \tag{4.3.5}$$

where T is a lower triangular matrix with positive diagonal elements, hence unique. In terms of this Cartesian reference frame, the data matrix is now transformed into

$$Z = Y(T')^{-1} \tag{4.3.6}$$

To find the "simple structure" subspaces, we look for unit vectors $\underline{t}$, such that

$$Z\underline{t} = \underline{v} \tag{4.3.7}$$

and $\underline{v}$ has the property that as many elements as possible are close to zero in the following sense:

Let $\underline{z}_i'$ denote the ith row of Z and v_i denote the ith element of $\underline{v}$. Then it is clear that

$$\underline{z}_i'\underline{t} = v_i \tag{4.3.8}$$

The ith element in $\underline{v}$, namely v_i, is considered close to zero if

$$v_i/\sqrt{\underline{z}_i'\underline{z}_i} < \sin \alpha/2 \tag{4.3.9}$$

where $\sin \alpha/2$ is determined by (3.4.1).

Let us now consider the significance of the expressions (4.3.8) and (4.3.9). $\underline{t}$ is a unit vector and so is $\underline{z}_i'/\sqrt{\underline{z}_i'\underline{z}_i}$. Their "inner product," $\underline{z}_i'\underline{t}/\sqrt{\underline{z}_i'\underline{z}_i}$, therefore is the cosine of the angle between the vectors $\underline{t}$ and $\underline{z}_i'$. But

$$\underline{z}_i'\underline{t}/\sqrt{\underline{z}_i'\underline{z}_i} = v_i/\sqrt{\underline{z}_i'\underline{z}_i} \qquad (4.3.10)$$

and hence $v_i/\sqrt{\underline{z}_i'\underline{z}_i}$ is the cosine of the angle between the vectors $\underline{t}$ and $\underline{z}_i'$. Since $\underline{t}$ is unit normal to the subspace (4.3.9) is established.

For a given number of experimental units, n, and p measurements on each of them, we can determine P_p using (3.4.6) and hence $\sin \alpha/2$ using (3.3.4). (4.3.9) is then a test to determine if the vector corresponding to a given data point (reduced in terms of z's) is close to the subspace to which $\underline{t}$ is orthogonal. If the vector (and hence the data point) is close to the subspace, we regard the point as falling in the overdetermined region, and treat the corresponding element of $\underline{v}$ as "zero." We can examine each element of $\underline{v}$ in this manner and determine how many "zero" elements are there. It is the count of these "zero elements" which we subject to the Beta test (3.4.4). If this test is significant, we say that the subspace is overdetermined and report it as a solution, provided it is not "close" to any subspace already found.

The transformation vectors $\underline{t}$ are found in a manner analogous to Thurstone's "Analytical Method" combined with his earlier "Single Plane Method" [36]. According to this method, each row of the reduced data matrix Z is used as a point of departure to find a vector $\underline{t}$. Let us start with the ith row vector $\underline{z}_i'$. We shall assume that $z_{i1}, z_{i2}, \ldots, z_{ip}$ are elements of $\underline{z}_i'$. Then $\underline{t}_o$, the initial approximation to $\underline{t}$ is obtained by normalizing $\underline{z}_i'$, i.e.,

$$\underline{t}_o' = \underline{z}_1'/\sqrt{\underline{z}_1' \underline{z}_1} = (t_{o1}, t_{o2}, \ldots, t_{op})$$

with this trial, the projections

$$\underline{v}_o = Z\underline{t}_o$$

are calculated and the elements v_{oi} are tested for closeness to zero. If a value is very close, a large weight is assigned to the point, for the subsequent weighted regression technique. If it is large, the weight may be zero. Following Thurstone, we use discrete (step) weights, as follows:

If $0 < v_{oi}/\sqrt{\underline{z}_1' \underline{z}_1} < \sin \alpha/2$

w_i = BND (BND, bound, initially 8).

If $\sin \alpha/2 < v_{oi}/\sqrt{\underline{z}_1' \underline{z}_1} < 2\sin \alpha/2$

w_i = BND-1

If $v_{oi}/\sqrt{\underline{z}_1' \underline{z}_1} >$ BND x $\sin \alpha/2$

$w_i = 0.$

With these weights we can follow a weighted regression scheme to obtain an improved vector $\underline{t}_1$. This can be further simplified by a relation (due to Thurstone) which expresses

$$t_{1j}^* = \frac{t_{oj} - u_j}{1 - t_{oj}u_j} \qquad (4.3.11)$$

where $u_j = \sum_{i=1}^{n} z_{ij} v_{oi} \sqrt{w_i} \Big/ \sum_{i=1}^{n} z_{ij}^2 \sqrt{w_i}$ . . . (4.3.12)

$t_{ij} = t_{ij}^*/\sqrt{\underline{t}_1^{*\prime} \underline{t}_1^*}$ are then

elements of the new trial vector $\underline{t}_1$. We calculate

$$\underline{v}_1 = Z\underline{t}_1$$

and once again the weights are assigned following the scheme detailed above.

However, before assigning these new weights the BND value is reduced to 3, in the next cycle to 2 and then kept at 1 for the remaining iterations. Beginning with $\underline{t}_0$, seven iterations are performed which give rise to $\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_7$. The last one, $\underline{t}_7$, is retained as $\underline{t}$ and $Z\underline{t} = \underline{v}$ is formed. If the number of "zero" entries in this $\underline{v}$ is significantly large as explained earlier, we will have determined a well defined subspace, to which $\underline{t}$ is normal. The entire process is repeated, with each row taken as a trial value. A given row may or may not identify a subspace. If it leads to an overdetermined subspace, the solution, except for the first one, is checked as explained in section 4.2 to ensure that the subspace is "different" from any of the ones already found from previous rows. For this purpose, we require that the cosine of the angle between two planes should not exceed 0.7 meaning that the planes were apart by at least 45°. Once again we would like to mention that this is an arbitrary requirement; we found this useful in our experimental studies.

Now we must retransform to the original data frame. The vectors $\underline{t}$ that we obtain above, are with reference to the matrix Z. Our original objective was to remove mean shifts and then look for experimental units which lie on subspaces. However, in working with Z, we have removed the mean shifts and also reduced the metric to an orthogonal Cartesian frame. This was a matter of convenience. To restore the original metric, we must obtain the solution in terms of Y. This can be easily obtained as under. (4.3.6) is the transformation of Y into Z and (4.3.7) is the simple structure solution in terms of Z. Substituting (4.3.6) into (4.3.7) we obtain

$$Y(T')^{-1}\underline{t} = \underline{v} \qquad (4.3.13)$$

from which we conclude that $(T')^{-1}\underline{t}$ is the transformation in terms of Y.

The computer program reports both $\underline{t}$ and $(T')^{-1}\underline{t}$. The vector $\underline{t}$ is reported under the heading "Vector which transforms original factor matrix into the above plane no. v_" and $(T')^{-1}\underline{t}$ is reported under the heading "Transformation vector to transfer raw data to simple structure." The corresponding elements of $\underline{v}$ are also reported. We shall also refer to elements of $\underline{v}$ as the "loadings" or "scores" of original points with reference to the simple structure plane determined.

For each $\underline{t}$, only the vectors $(T')^{-1}\underline{t}$ are conveyed to the display program, since we plot the original data points, in terms of the observed variables. As well be explained in Chapter 5, the display program takes one of the observed variables as one axis and some specified vector as another axis. For a given choice of a variable and a vector, the vector is reduced in such a manner that it forms an orthogonal frame of reference with the chosen observed variable. A question may be raised as to why one of the axes always corresponds to an observed variable. In this connection, it should be pointed out that a vector specified by the user, is equivalent to an artificial variable, a linear combination of the observable ones. The elements of the vector given by the user serve as weights in forming this linear combination. When we view the displays with reference to an orthogonal frame of reference consisting of an observable variable against an artificial variable, we can make an inference as to how an observable variable compares with an artificial variable. If both reference axes were to correspond to artificial variables, the displays would be hard to interpret in a realistic sense. This is our main consideration in insisting that one of the axes correspond to an observable variable. Further, if we should permit a user to select any two vectors, these could be translated

into an orthogonal frame of reference in many different ways leading to utter confusion.

4.4 Elimination of Variables

One notable difference between the rotational problem in factor analysis, and the problem presented in this dissertation, is the fact that the former focuses attention on those points (variables, in that case) which are far removed from the subspaces. By contrast, the latter pays attention to only those points which are so close to a subspace that they define it. It is these data points only which require one variable less for adequate description. It should be noted, again, that such a subspace is not a single region within the p-dimensional space. Rather, for each simple structure plane, there are as many (parallel) (p-1)-dimensional hyperplanes as there are clusters. Points which, in this sense, fall into the same subspace may be far removed from each other, inasmuch as they may be in different clusters. But even with their distinct neighbors, they share the property that the same (p-1) variables are sufficient to explain their characteristics. It is for this reason that we expect entirely new principles of classification of data points, different from what could be expected by varying or refining cluster analysis or factor analysis techniques.

Mathematically, we could use for description, (p-1) linear combinations of the original p variables, with the combinations chosen within the hyperplane orthogonal to the $(T^{-1})'\underline{t}$ vector. For real life interpretation, this procedure would be useless. Surely we are better off with p observable variables than with (p-1) artificial ones. The principle of parsimony is not just a principle restricted to dimensionality. It would

seem reasonable, then, to eliminate, for the data points close to one of the subspaces, that observable variable which contributes the least to the description of this selection of data points.

As a measure of proximity of each variable to the artificial variable which defines the subspace, we propose to use the correlation between the original variables and the artificial variable. This can be obtained as follows. Recall that Y is obtained from the original data matrix after subtraction of the appropriate cluster means. Hence

$$\underline{1}'Y = \underline{0}' \qquad (4.4.1)$$

where $\underline{1}'$ is a row vector consisting of all 1's and $\underline{0}'$ is a null row vector. Also,

$$\begin{aligned} \underline{1}'\underline{v} &= \underline{1}'Z\underline{t} \\ &= \underline{1}'Y(T')^{-1}\underline{t} \qquad \text{(By (4.3.6))} \\ &= \underline{0}' \qquad (4.4.2) \end{aligned}$$

and

$$\begin{aligned} \underline{v}'\underline{v} &= \underline{t}'Z'Z\underline{t} \\ &= n_e\underline{t}'I\underline{t} \\ &= n_e \qquad (4.4.3) \end{aligned}$$

since $\underline{t}$ is a unit vector. By virtue of the fact that $Z'Z = n_e I$, and because of (4.4.1) $(1/n_e)Y'\underline{v}$ will be an unbiased estimate of the covariances between the original variables and an artificial one on which the data points have scores which are the elements of $\underline{v}$. There will be as many $\underline{v}$ vectors as there are overdetermined subspaces (each corresponding to a different, but possibly overlapping, selection of data points). For each of these, we must determine its correlations with the original variables. Thus, if there are 4 variables and 3 different solutions (overdetermined subspaces), we will have a total of 4 x 3 = 12 correlations. Now, for a given $\underline{v}$ (a given subspace),

$$\begin{aligned} Y'\underline{v} &= Y'Z\underline{t} \\ &= Y'Y(T')^{-1}\underline{t} && \text{(By (4.3.6))} \\ &= n_e TT'(T')^{-1}\underline{t} && \text{(By (4.3.4) and (4.3.5))} \\ &= n_e T\underline{t} && (4.4.4) \end{aligned}$$

T is the same lower triangular matrix that was used in reducing the metric underlying the matrix Y to a Cartesian reference frame. From (4.4.4), we conclude that

$$(1/n_e)Y'\underline{v} = T\underline{t} \qquad (4.4.5)$$

Hence the estimated covariances between the original variables and a single artificial variable, are elements of $T\underline{t}$. To obtain the correlations, we have to divide each of these elements by the square root of the estimate of the variance of the artificial variable and the square root of the estimate of the variance of the observable variable. Using (4.4.2) and (4.4.3), we conclude that the estimate of the variance of the artificial variable is

$$(1/n_e)\underline{v}'\underline{v} = 1 \qquad (4.4.6)$$

since a sum of squares $(\underline{v}'\underline{v})$ must be divided by degrees of freedom to produce a variance estimate. Hence to obtain the correlations, we have to divide the elements of $T\underline{t}$ by the square root of the estimate of the variance of the observable variable only. In the computer program, after the vector $\underline{t}$ is obtained, the product $T\underline{t}$ is formed and the correlations are then calculated by division of each of the elements of $T\underline{t}$ by the square root of the estimate of the variance of the corresponding observable variable. The estimates of the variances of the observables are still available in the final step of the cluster analysis part of the program and these values are stored for use at this time. T is also stored.

Once the correlations between the original variables and their scores with reference to a subspace (i.e., vector $\underline{v}$) are available, it is easy to determine which variable should be discarded. In discriminant analysis, we come across the concept of correlations between original variables and the discriminant function. The discriminant function is nothing but a linear combination of the original variables which discriminates best between experimental units in a specified sense. The purpose for which we want to discriminate is important as the discriminant functions for different purposes are usually different. Given these things, the variable which correlates most strongly in absolute value with the discriminant function is the most important in discriminating. In our study the vector $\underline{t}$ which transforms the original observations into $\underline{v}$ plays a similar role in the sense that the elements of a well defined subspace represented by $\underline{v}$ has most elements near zero (see section 4.3) and a few far removed from zero. In analogy with discriminant analysis, $\underline{v}$ is the vector that produces the two groups of data points. Thus, the variable which correlates most strongly with this artificial variable, contributes most to the discrimination process. It is this maximally correlated observed variable which should be eliminated for, after it has been discarded the other variables contribute far less to the discrimination between these two sets of data points. If only one variable is to be sought which would explain the difference between these data points which fall into the subspace and those that do not, it would be this one which is closest (has highest correlation in absolute value) to the expendable artificial variable. Note the exact opposite of this technique to discriminant analysis, where we seek the best discriminator. Here we identify the "most expendable" artificial variable. By our correlational technique, we have identified,

for each subspace, that observable variable which is closest to the artificial variable not needed in the description of the selected data subset. Note again, the need for this correlational interpretation. If it were argued that the artificial variable itself ought to be discarded, we would be left with (p-1) artificial variables, a rather unsatisfactory situation. After the correlational approach, we have (p-1) observed variables left.

CHAPTER V

DESCRIPTIONS OF COMPUTER PROGRAMS

5.1 The Computer Programs

The algorithms explained in previous chapters have been synthesized into two computer programs which are available at the University of Georgia. The first program, named CLUSTR, and described in the next section, identifies the clusters and overdetermined subspaces. The second one is available as a conversational system for the IBM 2250 Graphics unit. In this chapter, we describe these two computer programs. The following chapter will contain instructions regarding the use of these programs, and the interpretation of graphical displays.

5.2 An Algorithm for Identification of Points Lying on a Subspace

In this program, beginning with the data matrix, we first identify the clusters. This is essentially the algorithm proposed by Bargmann and Graney [5]. However, the algorithm proposed by Bargmann and Graney stops at the identification process. Since we also need to identify points lying on an overdetermined subspace, we extend the algorithm further. A complete listing of the program is contained in Appendix A. This program can be logically divided into 2 parts. Up to statement number 811, it is essentially the reproduction of the program developed by Bargmann and Graney [5], where a complete documentation of this part can be found. We have made a small change in the program to suit our needs. As explained in Bargmann and Graney [5], their program makes three passes to identify

clusters. The passes made in their program are controlled by the statement immediately following statement number 444. In our program, CLUSTR, this has been replaced by the transition to the subspace-identification program. Further, the program developed by Bargmann and Graney [5] does not calculate cluster means or the "within" matrix after three passes. Their program computes these quantities at the beginning of the first pass, and after the first and second passes. We require these quantities for our search for the points lying on subspaces. These quantities are calculated in statements between number 811 and number 20001. At this stage, we also punch cards containing the means for each cluster and each variable. These will be needed prior to the execution of ELLIPSE described below. In statements between number 11020 and number 10001, we standardize the original points and reduce them to zero mean and a Cartesian metric. This, then, is the beginning of the second logical part of the program. Here, we determine the points lying on subspaces, using the method of weighted least squares together with Thurstone's "Analytical Method" and the "Single Plane Method." The algorithms (including the change of the BND variable) have been presented in sections 4.2 and 4.3

The output of this program consists of two parts--a print out and a punched deck. The printed output contains the results of three passes made to find clusters. The results listed after the third pass are the final results relating to cluster analysis. It shows which point belongs to which cluster. It also shows at what level the points got included in the cluster. The second part of the printed output gives various simple structure solutions. It gives vectors which transform the original observations into simple structure planes. For each of these vectors, the correlations between the original variables and the "scores" of points

with reference to this vector, the number of points falling into the simple structure plane corresponding to this vector, and the probability of these many points falling into this simple structure plane, are given. A simple structure plane is not included as a solution if the probability of the number of points falling into this plane is greater than 0.01 or if it is within 45° of a plane already found.

Apart from the printed output, a punched deck is also produced. These are data cards which are later loaded into data sets, as described below. The first few cards in this deck--equal in number to the number of clusters formed--are the cards containing cluster means. The next set of cards contains vectors which transform the original data points into simple structure solutions. The number of clusters is designated by NG and the number of simple structure solutions is designated by NSOL. This program also reproduces the cards for each data point with the following additional information. For each data point, the card contains a serial number in columns 1-3, the number of the cluster to which it belongs in column 4, and the simple structure planes in which it is included in columns 5-14. In column 4, a '1' is punched if the point belongs to cluster number 1, '2' if the point belongs to cluster number 2, etc. '0' is punched in column 4 if the point was not assignable to any of the clusters. The simple structure planes to which the point belongs is indicated in columns 5-14 as follows: A '1' in column 5 indicates that the point falls into simple structure number 1, a '1' in column 6 indicates that the point belongs to simple structure number 2, etc. (with provision for up to 10 solutions). This is certainly more than adequate capability. Zeros or blanks in columns 5 to 14 (the computer program punches zeros) indicate absence of this point in the corresponding simple structure plane. Columns 15-70

contain the original coordinates of each data point. The entire punched deck output is also printed out. The cluster means are printed at the end of the third pass, and the vectors which transform the raw data into simple structure solutions are printed immediately after the printout of the corresponding solution. The information punched into the last NP cards of the punched deck is also printed as a final summary. The user may find it helpful to keep this summary with him while he studies the displays.

5.3 Loading the Data Set (Utility Program IEBGENER)

After the user has subjected his data to the CLUSTR program, he should next run the IEBGENER program. This program supplies the output of CLUSTR program as input to the ELLIPSE program. A header card, containing the number of points, the number of variables, the number of groups, and the number of overdetermined subspaces identified by the CLUSTR program, is put before the punched deck produced by the CLUSTR program, and the IEBGENER program is executed. A sample deck set-up for the IEBGENER program is given in Chapter VI; this is a utility routine which transfers the cards to disk.

5.4 The Display Program and the Conversational System

The second program of the package serves to display projections of the clusters and subspaces, on an IBM 2250 Graphics Display Unit. It enables a user, at the console, to communicate with the system and to manipulate displays appearing on the scope. The program, named ELLIPSE, is capable of handling up to 8 variables, 50 data points, 10 simple structure solutions and 5 groups or clusters. The numbering of the

variables is implicit. The first coordinate, for each data point, is regarded as variable number 1, the second coordinate as variable number 2, etc.

The program is an interactive one in the sense that the user, at the console, can decide on variations for later displays on the basis of what he saw in the earlier ones. The input of the data is so formatted and programmed that, if a user wishes to reassign a point from one cluster to another, or if he wishes to change cluster centers, etc., he only needs to make a change to this effect in the corresponding data cards. This capability gives the user an opportunity to redefine clusters and subspaces on the basis of the displays generated. The interaction between the user and the system is achieved through the use of the programmed function keys and the alphameric keyboard which is a part of the 2250 Graphics Display Unit.

The program includes a main program and 8 subroutines. The main program calls two subroutines CALC and EXIBIT. CALC reads the entire input into the ELLIPSE program. The input consists of a header card containing the number of points, the number of variables, the number of clusters and the number of simple structure solutions; cluster means for each variable; vectors which transform the original data points (raw data) into various simple structure solutions; and the data points, together with information regarding the cluster to which a data point belongs and whether or not it lies on a given overdetermined subspace. As explained earlier, this input is given to the program through the execution of the IEBGENER Utility routine. The first (executable) statement of the CALC subroutine reads the header card. Following this, the DO 14 loop reads cluster means, the DO 114 loop reads the transformation vectors and the

DO 10 loop reads the data points. The array IG is used to store information regarding the cluster to which a data point belongs. Recall that for each data point, columns 5-14 contained '1' to indicate the corresponding simple structure plane to which this point belongs. This configuration of '1's and '0's is read as a 10 digit integer number and stored in the first column of the two-dimensional array INNSOL. The serial number of a data point is stored in the second column of INNSOL. In the DO 422 loop, the array IG is examined to determine the size of each cluster. The DO 429 loop, then, determines the total number of points assigned to clusters. The subroutine COR2 is now called which yields the within sum of squares and products matrix based on all the points belonging to clusters. The upper triangular part of this symmetric matrix is stored, columnwise, in the array DSPROD. In the DO 434 loop, each element of the array DSPROD is divided by n_e, the degrees of freedom, to yield an estimate S of the common variance-covariance matrix, Σ. Immediately following the statement number 434, SINV, a sub-routine from the IBM Scientific Subroutine Package, is called to invert the matrix S. The inverse of this matrix S, stored as the first column of the two-dimensional array A, is the metric (based on all points belonging to clusters) employed for unit ellipsoids around the clusters. The subroutnien CALC then calculates metrics corresponding to overdetermined subspaces. For each of the overdetermined subspaces, the DO 426 loop calculates a metric corresponding to each of the overdetermined subspaces on the basis of the points belonging to it. The inner DO 427 loop examines the 10 digit integer numbers stored in the first column of the array INNSOL to determine the points lying on a particular subspace. For each subspace, the subroutine

COVIS is utilized to calculate a new within-cluster sum of squares and products matrix. In the DO 442 loop, each element of this matrix is divided by the number of points belonging to the overdetermined subspace, to yield an estimate of the variance-covariance matrix based on the points belonging to the subspace only. The subroutine SINV is then called to invert this matrix. The inverse matrix is used as the metric for ellipsoids corresponding to the overdetermined subspace. The metric corresponding to the overdetermined subspace number 1 is stored as the second column of the two-dimensional array A, the metric corresponding to the overdetermined subspace number 2 is stored as the third column of the array A, etc. This is accomplished in the DO 433 loop. The DO 15 loop, beginning at statement number 450, calculates the maximum and minimum values for each variable. These maximum and minimum values are required later for scaling purposes to accommodate each of the projected data points within the screen limits. A flow chart for the subroutine CALC is given in figure 5.4.1.

Subroutine EXIBIT

The subroutine EXIBIT begins with a call to the DISPLA subroutine of the GRAF (Graphics Addition To Fortran) package [16]. This results in setting up GDSX, GDSY, GTEXT, GPOINT, GDSE, GDSER and GINPUT as display variables. The subroutine LIGHTS of the GRAF package is next called to turn on the lights corresponding to the programmed function keys numbered 1 up to the number of variables, keys 27 to 29 and 31. After these preliminaries, the subroutine MESSGE is called to display an informative message about the program, for the benefit of the user. The subroutine MESSGE returns the control to the subroutine EXIBIT as soon as the user

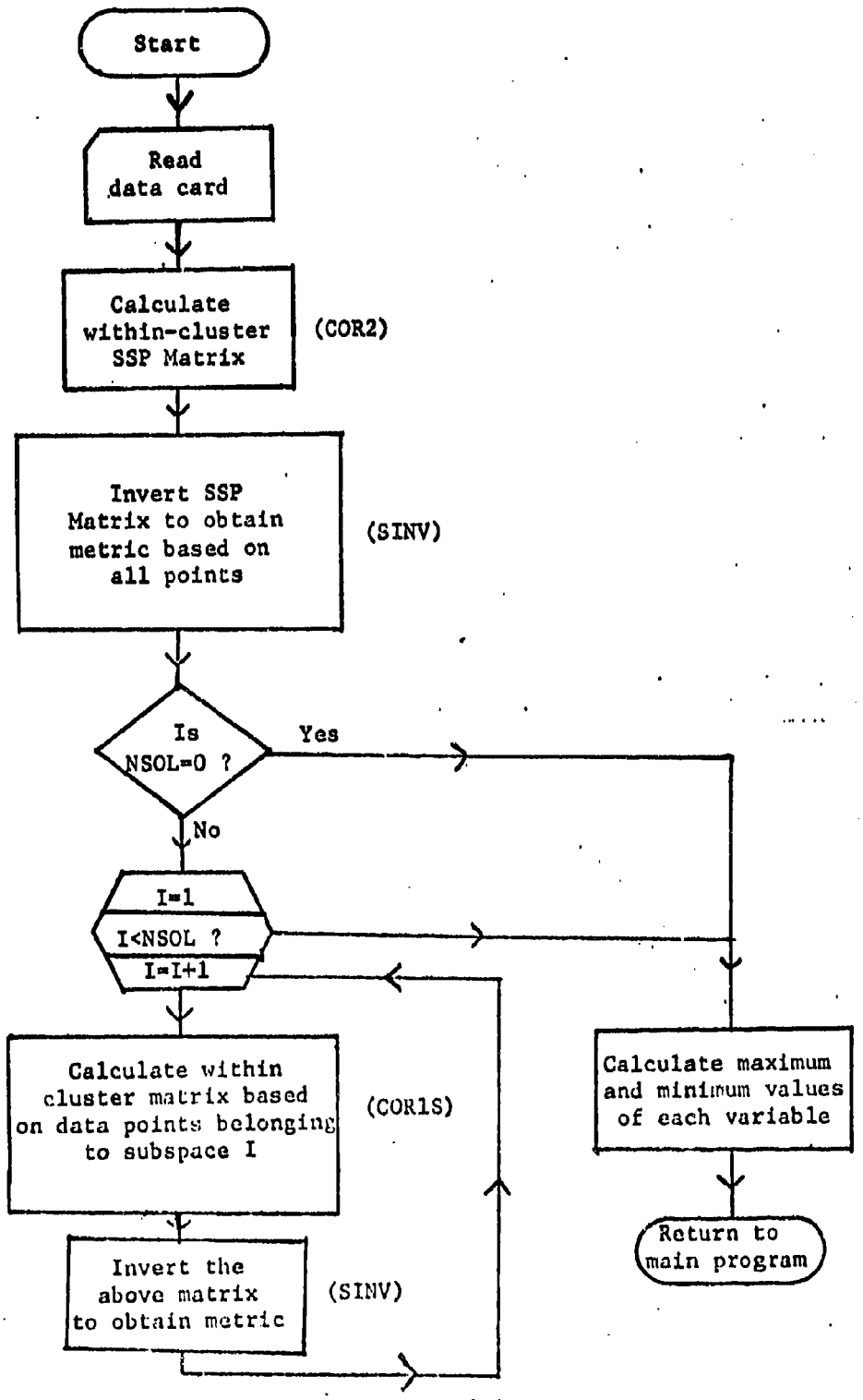
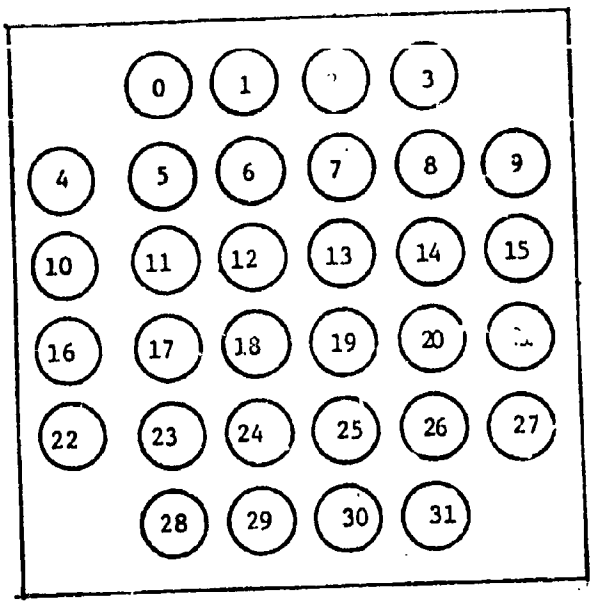
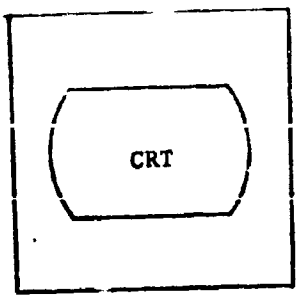

Start
Read data card
Calculate within-cluster SSP Matrix
(COR2)
Invert SSP Matrix to obtain metric based on all points
(SINV)
Is NSOL=0 ?
Yes
No
I=1
I<NSOL ?
I=I+1
Calculate within cluster matrix based on data points belonging to subspace I
(COR1S)
Invert the above matrix to obtain metric
(SINV)
Calculate maximum and minimum values of each variable
Return to main program

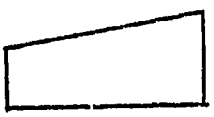
Figure 5.4.1

presses any programmed function key. The message appearing on the screen is erased, a variable NTEST (used later to register the depressed key) is set initially equal to 1 and a variable NGG (a flag which is used to indicate change of vectors) is set equal to 0 and the control goes to statement number 20. This statement is a call to the subroutine KEYIN, with NTEST, the input argument having been set equal to 1. The subroutine KEYIN accepts from the user a vector and the number of a variable, in order to form a 2-dimensional plane. The user indicates his choice of the number of the variable by pressing the corresponding programmed function key. The variable NAXIS is used as an output argument of the subroutine KEYIN and on return contains the number of the programmed function key pressed by the user. As will be seen later, on return from KEYIN, the variable NAXIS must either have a value equal to the number of the variable the user wishes to utilize, or 30. If NAXIS equals 30, it is implied that the user wishes to stop and the display program comes to an end. Otherwise, the subroutine ELLPSE is called with the current value of NAXIS (the number of the variable) as input argument. The subroutine ELLPSE displays the projections of the original data points, and the unit ellipsoids having the metric based on all the points belonging to clusters, onto the 2-dimensional plane formed by the vector and the variable supplied by the user, provided there is no singularity or redundancy involved (see ELLPSE below). After the above displays appear, the user is expected to press a programmed function key. If he presses key 29 or 31, the control comes to statement number 80. This results in erasing the current displays and then a call to the subroutine KEYIN, with NTEST, the input argument, being 29 or 31. As before, the call to the subroutine KEYIN is followed by a call to the subroutine ELLPSE and the cycle



Programmed Function Keyboard





Alphameric Keyboard

IBM 2250 Display Unit

continues. If a key corresponding to the number of an overdetermined subspace was pressed, the control comes to statement number 61, and if key 30 was pressed, the display program comes to an end. If the control comes to statement number 61, the subroutine RELPSE is called to display the projections of the overdetermined subspace. If none of the above mentioned keys is pressed, an error message, as contained in the format statement number 62, appears on the screen. The error message continues to appear until the user presses a proper key or, in case of singular or redundant situations, rectifies the situation. If the subroutine RELPSE is called, after the projections of the overdetermined subspace are displayed, the user is expected to press a programmed function key. Once again, if key 29 or 31 is pressed, the current displays are erased, the control comes to statement number 20 and the subroutine KEYIN is called. The program terminates if key 30 was pressed. If the key corresponding to the number of the overdetermined subspace was pressed, that part of the current display pertaining to the projection of the overdetermined subspace is erased, and the subroutine RELPSE is called to display the projections of the overdetermined subspace that is now being requested. An error message appears if none of the abovementioned keys is pressed, and the cycle continues in this manner. A flowchart of the subroutine is given in figure 5.4.2.

Subroutine KEYIN

The subroutine KEYIN accepts a vector and the number of the variable from the user, to form a 2-dimensional plane. It has one input argument, NTEST, and an output argument, NAXIS. The first statement in the subroutine tests the value of NTEST. If it equals 31, that part of

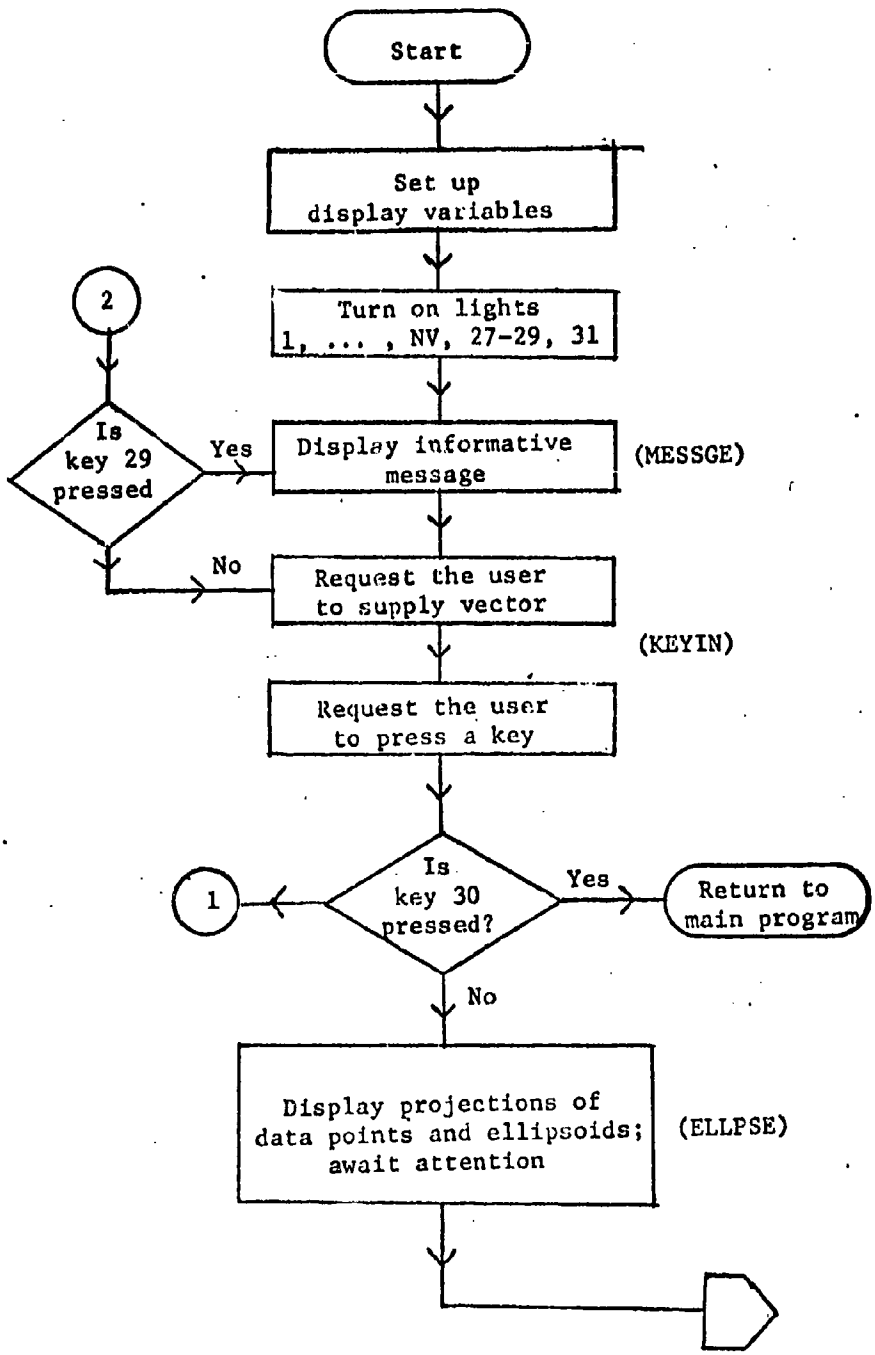

Start
Set up display variables
2
Turn on lights 1, ... , NV, 27-29, 31
Is key 29 pressed
Yes
Display informative message
(MESSGE)
No
Request the user to supply vector
(KEYIN)
Request the user to press a key
1
Is key 30 pressed?
Yes
Return to main program
No
Display projections of data points and ellipsoids; await attention
(ELLPSE)

Figure 5.4.2a

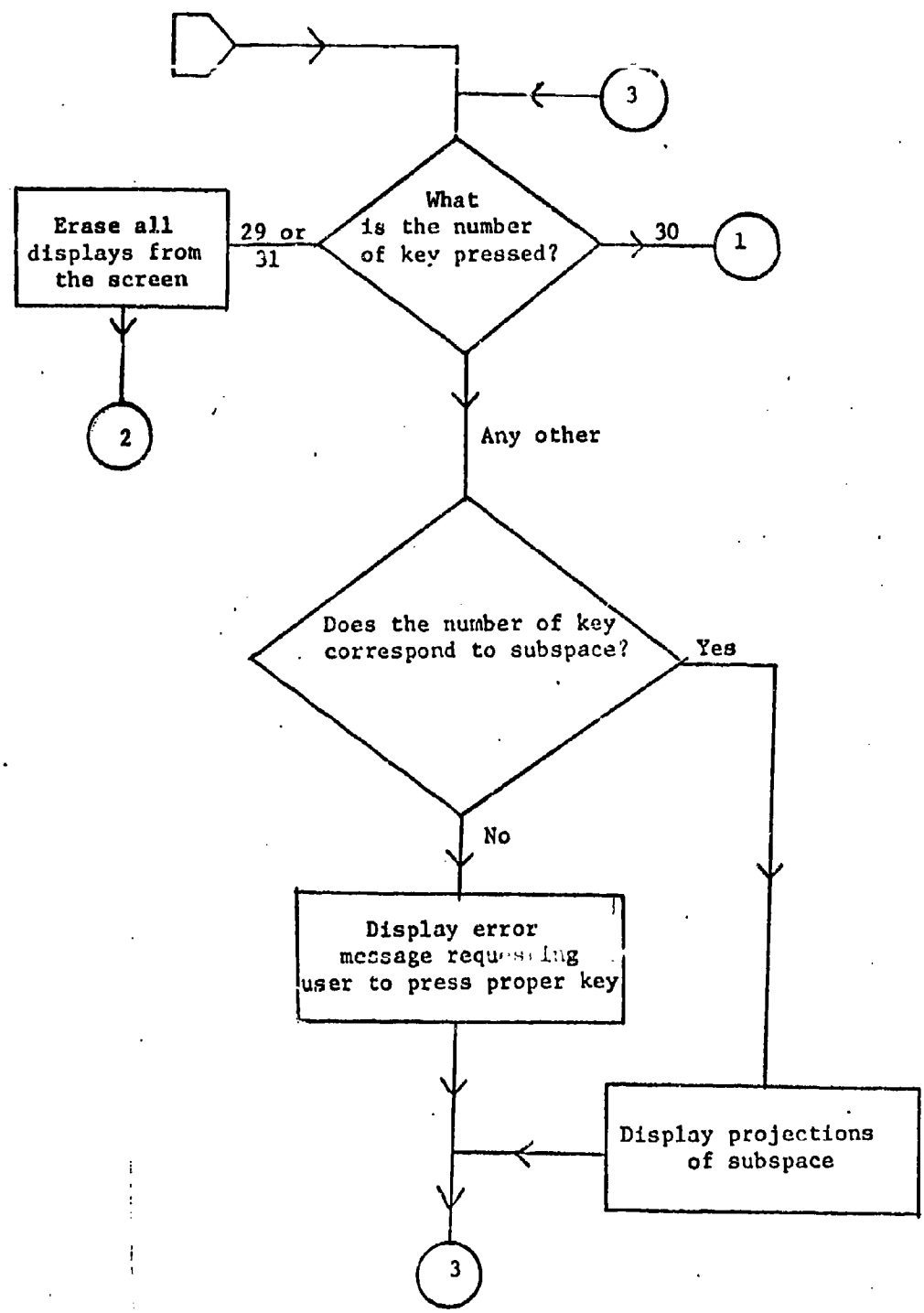

3
What is the number of key pressed?
Erase all displays from the screen
29 or 31
30
1
2
Any other
Does the number of key correspond to subspace?
Yes
No
Display error message requesting user to press proper key
Display projections of subspace
3

Figure 5.4.2b

the subroutine which accepts a vector from the user is skipped, and the control comes to statement number 101. Otherwise, the control comes to statement number 99, and whatever appears on the screen is erased to prepare to accept a vector from the user. On the first call to the subroutine KEYIN, the input argument, NTEST, is set equal to 1 so that the control invariably comes to statement 99. On subsequent occasions, however, NTEST will have a value of 29 or 31. The DO 110 loop displays the transformation vectors suggested by the rotation (CLUSTR) program for the information of the user. The message contained in the format statement number 3000, requesting the user to supply a vector, then appears on the screen. The program now awaits the user to supply the vector. The calls to SCTDV and DVTDM subroutines of the GRAF package transfer the vector supplied by the user from the screen to the display variable table and from there to the dummy unit 4. The vector is read from the dummy unit 4 into the array RINPUT. Before, however, the vector is read, the DO211 loop transfers the current values stored in the RINPUT array to the RWRKNG array. This is done to insure that the vector previously supplied by the user is not destroyed in case he wants to continue with that vector. The DO 213 loop checks if the user supplied a null vector. If so (as is the case when he just wants to continue with the previous vector), this is always replaced by the previous non-null vector as would be apparent from the DO 216 loop. The only exception, as will be seen from the statement following the statement number 213, is the first call to the subroutine. This would result in singularity and the user will receive an error message instead of the displays.

After the vector is accepted, the control comes to statement number 101. The message contained in the format statement number 1658 appears on the screen. The loop beginning with the statement number 60 insures that no value other than 30, or the number corresponding to the variable which the user wishes to utilize, will be returned as the value of the output argument NAXIS. The user can, of course, go back to statement 99, the beginning of the program, if he presses key 28. This gives him a chance to amend the vector already supplied. A flowchart of the subroutine is given in figure 5.4.3.

Subroutine ELLPSE

This subroutine is used to display projections of original data points, and the unit ellipsoids having the metric based on all points belonging to clusters, onto the 2-dimensional plane formed by the vector and the variable supplied by the user. The DO 10 and DO 11 loops set up a matrix R formally given by

$$R = \begin{bmatrix} 0 & a_1 \\ 0 & a_2 \\ \vdots & \vdots \\ 1 & 0 \\ \vdots & \vdots \\ 0 & a_p \end{bmatrix} \qquad (5.4.1)$$

where 1 in the first column appears in the row corresponding to the number of the variable chosen by the user, and $(a_1, a_2, \ldots, 0, \ldots, a_p)$ is the vector supplied by the user and modified to form an orthogonal axis

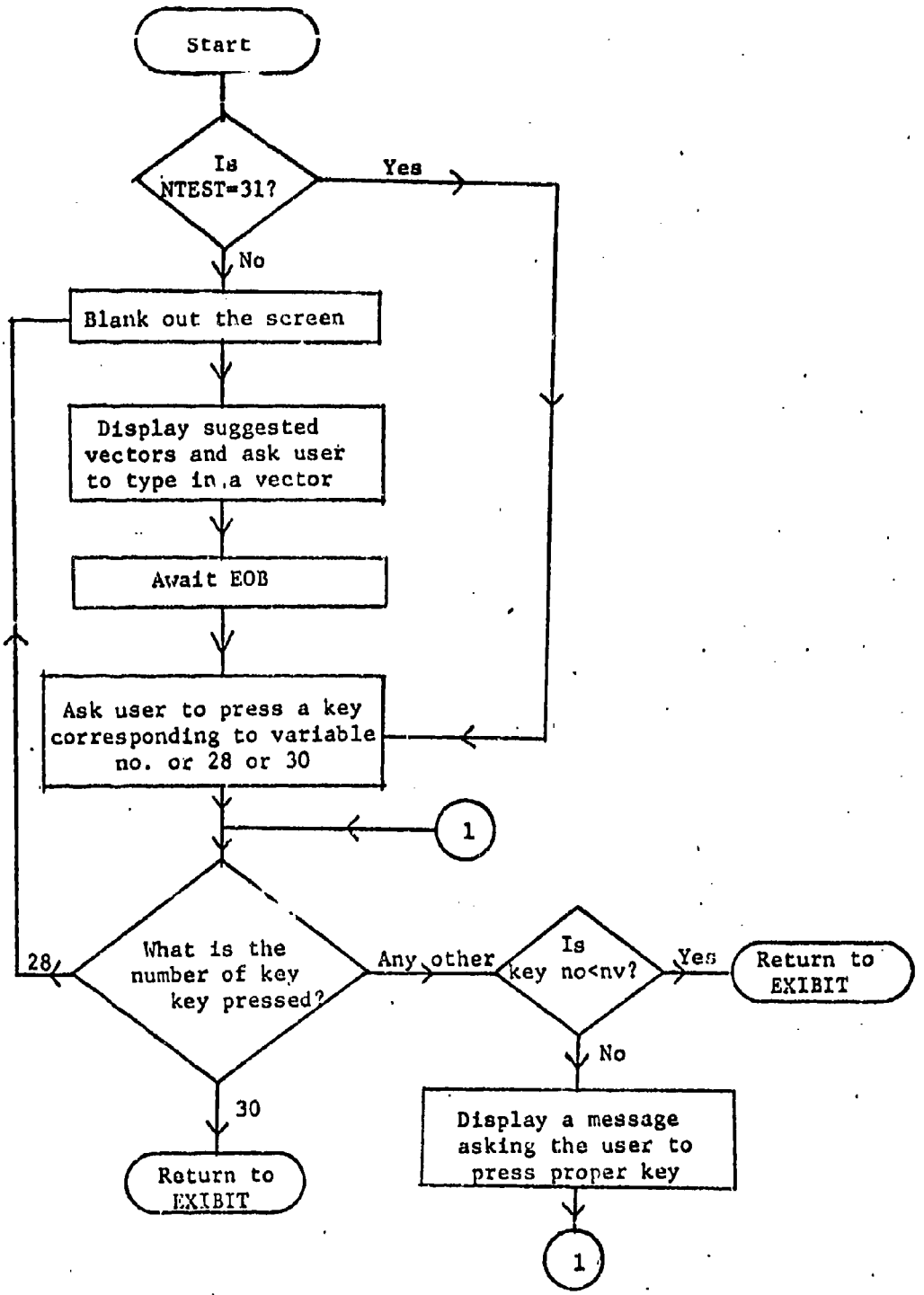

Start
Is NTEST=31?
Yes
No
Blank out the screen
Display suggested vectors and ask user to type in a vector
Await EOB
Ask user to press a key corresponding to variable no. or 28 or 30
1
What is the number of key key pressed?
28
Any other
Is key no<nv?
Yes
Return to EXIBIT
No
Display a message asking the user to press proper key
1
30
Return to EXIBIT

Figure 5.4.3

with the desired variable. If the user supplies a vector collinear with this choice of the observed variable, the second column of the matrix R will have all zero elements. The DO 20 loop, and the statement immediately following this loop, check for the above-mentioned possibility of singularity. If no singularity is present, the control comes to statement number 211; otherwise, the error message, as contained in the format statement number 1659, is displayed on the screen, and control is returned to the subroutine EXIBIT. At statement 211, the DO 21 loop is set up to normalize the second column of the matrix R. The DO 12 loop picks up the metric S^{-1} based on all points belonging to clusters and the subroutines MPRD and GTPRD of the IBM Scientific Subroutine Package are called, to form the matrix product $R'S^{-1}R$. The DO 13 loop, and the statement immediately following it, calculate the matrix product XR, where X is the matrix of original data points. For a projected data point, the element of the first column of the matrix is treated as its x-coordinate and the element of the second column is treated as y-coordinate. The DO 110 loop creates orders to plot points with these sets of x and y coordinates. The actual plotting is done by displaying, on the screen, at the place where the point should appear, its serial number, so that the user may know which data point projects into what region of the 2-dimensional display.

In the statements immediately following the DO 110 loop, the semi-axes of the ellipses to be displayed are calculated. They are based on the metric $R'S^{-1}R$ (of the projected ellipsoids). The cluster centers (centers of ellipses) are likewise transformed into $\underline{\mu}'_i R$. The points describing the circumference of the ellipses

$$(\underline{x}' - \underline{\mu}_1 R)\ R' S^{-1} R\ (\underline{x} - R'\underline{\mu}_1) = 1 \qquad (5.4.2)$$

where $\underline{x}' = (x_1, x_2)$ are the running coordinates, are constructed by angular sweep. Beginning with an initial angle of 5° the DO 150 loop calculates 72 different points describing the circumference of the ellipses. The DO 100 loop creates orders to plot these points and the ellipses appear on the screen when the statement immediately following the DO 100 loop is executed. A flowchart of the subroutine is given in figure 5.4.4

Subroutine RELPSE

This subroutine is used to display projections of ellipsoids having metric based on the points belonging to the overdetermined subspace being superimposed. If the metric for the ith subspace is denoted by S_i^{-1}, the subroutine calculates the points describing the circumference of ellipses

$$(\underline{x}' - \underline{\mu}_i' R)\ R' S_i^{-1} R\ (\underline{x} - R'\underline{\mu}_1) = 1 \qquad (5.4.3)$$

where R is as defined in ELLPSE. The technique employed to calculate these points is similar to the one employed before. Like ELLPSE, RELPSE also checks for singularity. If it is present, no displays of ellipses appear. It is to be noted that, if the user chooses one of the vectors suggested to him in the display (the vectors identified in the CLUSTR program), the ellipses constructed in RELPSE, if the user depresses the corresponding key, is quite flat, as intended. It is conceivable that, in such an instance, singularities could occur (though we did not see any in our examples) and hence the program checks for them.

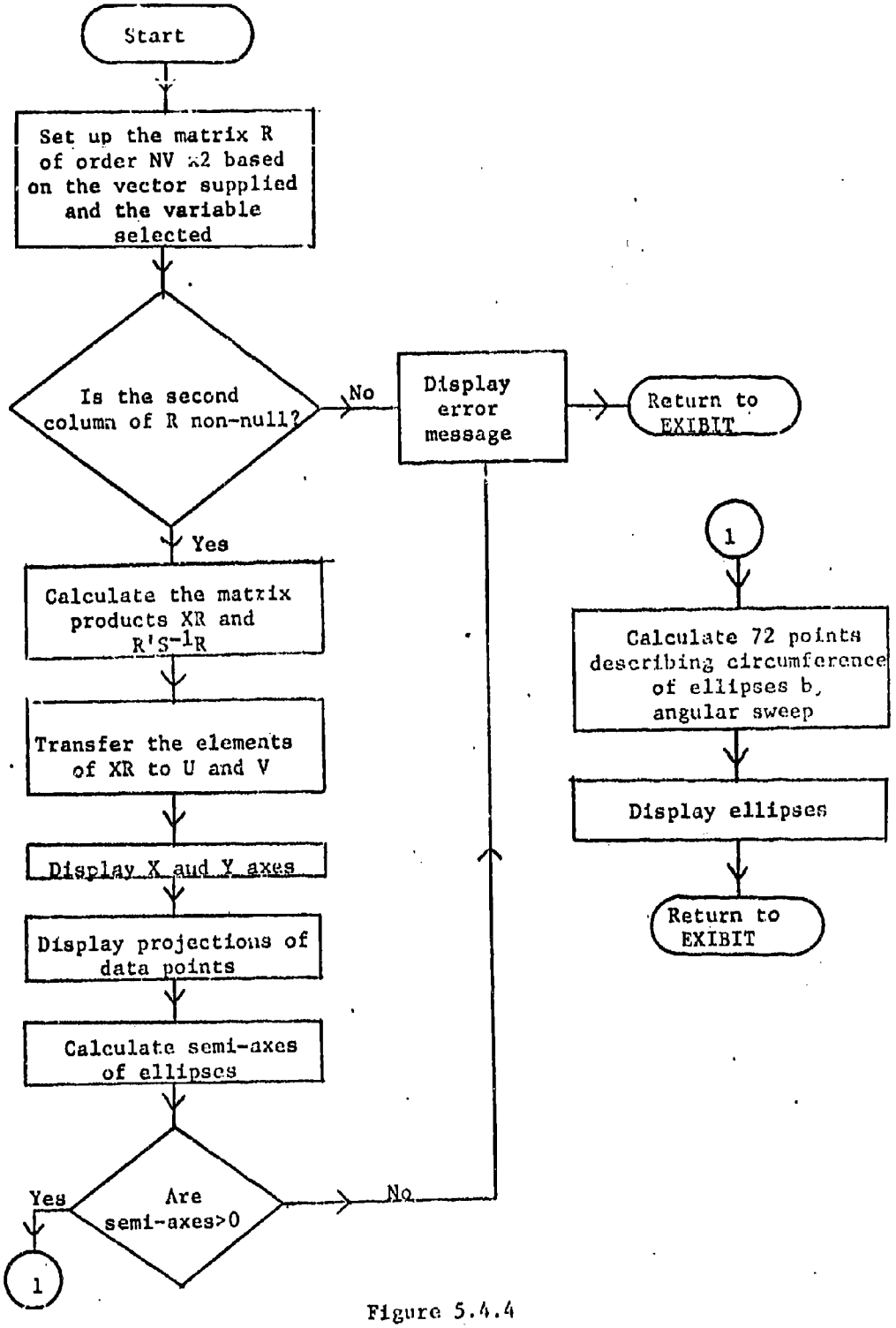
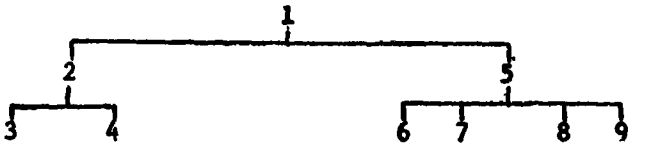

Start
Set up the matrix R of order NV x2 based on the vector supplied and the variable selected
Is the second column of R non-null?
No
Display error message
Return to EXIBIT
Yes
Calculate the matrix products XR and R'S-1R
Transfer the elements of XR to U and V
Display X and Y axes
Display projections of data points
Calculate semi-axes of ellipses
Are semi-axes>0
Yes
No
1
1
Calculate 72 points describing circumference of ellipses by angular sweep
Display ellipses
Return to EXIBIT

Figure 5.4.4

Overlay Structure

To reduce storage requirements, an overlay structure was designed as follows:



1. (Root) contains the main program and Function KD.
2. Contains subroutine CALC
3. Contains subroutine COR1S
4. Contains subroutine COR2
5. Contains subroutine EXIBIT
6. Contains subroutine MESSGE
7. Contains subroutine KEYIN
8. Contains subroutine RELPSE
9. Contains subroutine ELLPSE

Segment 1, along with the main program and function KD, also contains the system support routines IBCOM#, ARIH#, FIOCS#, ADCON#, and system utilities IHCUATBL, IHCUOPT and IHCTRCH. Segment 5 contains all of the GRAF routines required except BUFRS, CUR$$, RCUR$, READSC, SCNDVDK and SCTDV, which are included in segment 7. With the help of this overlay structure and equivalencing of a few arrays in the ELLPSE subroutine, it was possible to reduce the storage requirements to 64K bytes.

Deck Layout for ELLIPSE

```
//STEP1 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(ROOT),SPACE=(TRK,(150,10,5)),          C
              UNIT=SYSDA,DISP=(NEW,PASS)

//FORT.SYSIN DD *
```

```
Main program here
/*
//STEP5 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(LINKA3),DISP=(MOD,PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Function KD Source Deck
/*
//STEP2 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(LINKA),DISP=(MOD.PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Subroutine CALC Source Deck
/*
//STEP7 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(LINKA5),DISP=(MOD,PASS),UNIT=SYSDA
//FOKT.SYSIN DD *
Subroutine COR1S Source Deck
/*
//STEP6 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(LINKA4),DISP=(MOD,PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Subroutine COR2 Source Deck
/*
//STEP4 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(LINKA2),DISP=(MOD,PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Subroutine EXIBIT Source Deck
```

```
Subroutine EXIBIT Source Deck
/*
//STEP9 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&(CHAIN(LINKA8),DISP=(MOD,PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Subroutine MESSGE Source Deck
/*
//STEP10 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(LINKA9),disp=(MOD,PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Subroutine KEYIN Source Deck
/*
//STEP3 EXEC FORTGC
//FORT.SYSLIN DD DSNAME==&&CHAIN(LINKA1),DISP=(MOD,PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Subroutine ELLPSE Source Deck
/*
//STEP8 EXEC FORTGC
//FORT.SYSLIN DD DSNAME=&&CHAIN(LINKA6),DISP=(MOD,PASS),UNIT=SYSDA
//FORT.SYSIN DD *
Subroutine RELPSE Source Deck
/*
//S10 EXEC LKED,PARM=(LET,LIST,OVLY,XREF)
//LKED.SYSLMOD DD DSN=SYS1.GRAPHLIB(ELLIPSE),DISP=SHR,                    C
//               SPACE=(TRK,(0,0)
//LKED.SYSLIB DD DSN=SYS1.GRAFLIB,DISP=SHR
```

```
// DD DSN=SYS1.UGALIB,DISP=SHR
// DD DSN=SYS1.SSPLIB,DISP=SHR
// DD DSN=SYS1.FORTLIB,DISP=SHR
// DD DSN=SYS1.LINKLIB,DISP=SHR
// DD DSN=SYS1.GRAPHLIB,DISP=SHR
//LKED.MODULE DD DSN=&&CHAIN,DISP=OLD
//LKED.SYSIN DD *
 INCLUDE MODULE(ROOT)
 INCLUDE MODULE(LINKA3)
 INCLUDE SYSLIB(IBCOM#)
 INCLUDE SYSLIB(ARITH#)
 INCLUDE SYSLIB(FIOCS#)
 INCLUDE SYSLIB(ADCON#)
 INCLUDE SYSLIB(IHCUATBL)
 INCLUDE SYLIB(IHCUOPT)
 INCLUDE SYSLIB(ERRMON)
 INCLUDE SYSLIB(IHCTRCH)
 OVERLAY ONE
 INCLUDE MODULE(LINKA)
 INCLUDE SYSLIB(SINV)
 INCLUDE SYSLIB(MFSO)
 OVERLAY TWO
 INCLUDE MODULE (LINKA5)
 OVERLAY TWO
 INCLUDE MODULE (LINKA4)
 OVERLAY ONE
 INCLUDE MODULE (LINKA2)
```

```
INCLUDE SYSLIB(GAFERR)
INCLUDE SYSLIB(LIGHTS)
INCLUDE SYSLIB(WRFMT$)
INCLUDE SYSLIB(DETEKT)
INCLUDE SYSLIB(PLOT)
INCLUDE SYSLIB(DETAIN)
INCLUDE SYSLIB(DISPLA)
INCLUDE SYSLIB($$OVER)
INCLUDE SYSLIB(CHAR)
INCLUDE SYSLIB(POINT)
INCLUDE SYSLIB(LINE)
INCLUDE SYSLIB(PLACE)
INCLUDE SYSLIB($$$$BT)
INCLUDE SYSLIB($$INIT)
INCLUDE SYSLIB(DUMMY$)
INCLUDE SYSLIB($VOVER)
INCLUDE SYSLIB(CLOSE)
INCLUDE SYSLIB(LINE$$)
INCLUDE SYSLIB(UNPLOT)
INCLUDE SYSLIB(PLACE$)
INCLUDE SYSLIB(POINT$)
INCLUDE SYSLIB(ERASE)
INCLUDE SYSLIB(BLANK)
INCLUDE SYSLIB(RESET)
OVERLAY TWOA
INCLUDE MODULE (LINKA8)
OVERLAY TWOA
```

```
INCLUDE MODULE (LINKA9)
INCLUDE SYSLIB(SCNDVDK)
INCLUDE SYSLIB(CUR$$)
INCLUDE SYSLIB(BUFRS)
INCLUDE SYSLIB(SCTDV)
INCLUDE SYSLIB(RCUR$)
INCLUDE SYSLIB(READSC)
OVERLAY TWOA
INCLUDE MODULE (LINKA6)
OVERLAY TWOA
INCLUDE MODULE (LINKA1)
```

CHAPTER VI

USER'S GUIDE

6.1 Introduction

The user who is interested in using the programs described in the previous chapter would find himself in one of the following situations:

(i) He may not have analysed his multivariate data and may not yet have identified clusters and subspaces. If so, he should first subject his data to the CLUSTR program. The next section contains instructions on how to use (execute) this program.

(ii) He may have analysed his data using the CLUSTR program but has not loaded the data set for the ELLIPSE program. If so, he should execute the IEBGENER Utility routine and load the data set. This is described in section 6.3.

(iii) Finally, the user may have gone through the steps (i) and (ii) above and may want to see the displays of projected clusters and subspaces. The use of the ELLIPSE program including an indication of what to look for in the displays is described in section 6.4.

Section 6.5 contains an illustration.

6.2 The CLUSTR program

The analysis of the user's multivariate data begins with the identification of clusters and overdetermined subspaces, if any. For this purpose, the user must first analyse his data using the CLUSTR program.

This is a batch program. The following data cards need to be supplied:

Data Card 1

Columns 1-3, number of points (individuals or experimental units)

Columns 6-7, number of variables (responses) measured on each experimental unit

Columns 8-11, alpha level for cluster core on first pass (suggested value 0.90)

Columns 13-16, alpha level for cluster extension on first pass (suggested value 0.50)

Columns 18-21, alpha level for cluster core on second pass (suggested value 0.90)

Columns 23-26, alpha level for cluster extension on second pass (suggested value 0.50)

Columns 28-31, alpha level for cluster core on third pass (suggested value 0.90)

Columns 33-36, alpha level for cluster extension on third pass (suggested value 0.50)

Data Card 2

This is a variable format card and should contain the FORMAT by which each experimental unit is to be read.

The remaining data cards contain the observations, one card (or record which may consist of several cards) contains the coordinates of one point. The numbering of these points is implicit, according to the sequential order of these cards.

Our suggestion above that 0.90 should be used as alpha level for cluster core and 0.50 as alpha level for cluster extension is empirical. Of course, he can use any other set of values. For a detailed discussion of this matter the reader is directed to Craney [10].

The output of this program has been discussed at length in section 5.2 The punched deck produced by this program is required for the ELLIPSE program.

6.3 IEBGENER Utility routine

As explained in section 5.3, it is necessary to execute the IEBGENER Utility routine to load the output of the CLUSTR program into a data set required as input to the ELLIPSE program. The deck set-up for the execution of this utility routine is as follows:

(i) JOB card

(ii) //STEPG EXEC PGM=IEBGENER

(iii) //SYSPRINT DD SYSOUT=A

(iv) //SYSIN DD DUMMY

(v) //SYSUT2 DD DSN=SYS1.R2250,VOL=SER=UGA231,DISP=SHR,UNIT=2314

(vi) //SYSUT1 DD DATA,DCG=(RECFM=FB,LRECL=80,BLKSIZE=320)

(vii) Data Cards

(viii) /*

The data cards consist of a header card followed by the punched deck produced by the CLUSTR program (of course, the user could produce his own data cards, and any assignment of points to clusters or subspaces which he desires. In this respect the CLUSTR program is merely intended to give him guidance--but a very strong one indeed). The header card is made up as follows (all numbers right justified);

Columns 1-4, numbers of points (individuals or experimental units)

Columns 5-8, number of variables (responses)measured on each experimental unit

Columns 9-12, number of clusters identified by the CLUSTR program

Columns 13-16, number of overdetermined subspaces (simple structure solutions) identified by CLUSTR program

After the IEBGENER Utility routine is executed, one is ready for the ELLIPSE program.

6.4 The ELLIPSE program

This program works under the control of GMS (Graphics Monitor System). For greater detail on the operation of GMS see Penn [31]. In order to be operational under the conversational GMS, the load module of the program has to be a member of the partitioned data set GRAPHLIB. The user should verify, by typing the command $NAMES on the console typewriter, that the program, in fact, is a member of the GRAPHLIB data set. If not, the user will first have to compile and link edit the program using the deck set-up given in section 5.4. The user can then execute the program using the command $LINK ELLIPSE to link to it.

Photographs made during the use of the program are reproduced here. The user will find it helpful to refer to them while studying the rest of the section. Some of the photographs will be specifically discussed in the next section.

The execution of ELLIPSE begins with the display of an informative message. The user should carefully read the message. (Note especially the use of the programmed function key 30. This is to be pressed only when it is desired to stop the execution of the program.) The user should then press any key other than 0 or 30. He is now asked to type the coordinates of a vector which he wishes to utilize in order to form a

THIS PROGRAM DISPLAYS CLUSTERS BY PROJECTING THEM ON VARIOUS 2-DIMENSIONAL SUBSPACES.SIMPLE STRUCTURE SOLUTIONS CAN ALSO BE INDICATED.REFER TO YOUR INSTRUCTION CHART.HAVE YOU ENTERED ALL NECESSARY DATA VIA IEBGENER?

THE BOTTOM ROW OF THE PROGRAM FUNCTION KEYS IS LIT UP.THEY WILL BE USED AS DIRECTED.THE DARK ONE,KEY NO. 30,IS THE PANIC BUTTON.IT WILL RETURN YOU TO THE MONITOR.

IN CASE OF PANIC PRESS KEY 30

NOW PRESS ANY KEY TO GET STARTED

```
   ACCORDING TO THE ROTATION PROGRAM THE
FOLLOWING VECTORS TRANSFORM RAW DATA INTO
SIMPLE STRUCTURE SOLUTIONS NO.1 TO  3
 -0.301-0.275 0.854-0.309
  0.012 0.843-0.155 0.516
 -0.306-0.259-0.316 0.860

ENTER A TRANSFORMATION VECTOR,
NO MORE THAN 7 CHARACTERS,DECIMAL POINT
MUST BE TYPED
DEPRESS JUMP KEY AFTER EACH COORDINATE
X(1)=_
X(2)=
X(3)=
X(4)=
X(5)=
X(6)=
X(7)=
X(8)=
!F PREVIOUS VECTOR OK,JUST PRESS EOB
```

```
   ACCORDING TO THE ROTATION PROGRAM THE
FOLLOWING VECTORS TRANSFORM RAW DATA INTO
SIMPLE STRUCTURE SOLUTIONS NO.1 TO  3
 -0.309-0.275 0.856-0.309
  0.012 0.843-0.155 0.516
 -0.306-0.259-0.316 0.860

ENTER A TRANSFORMATION VECTOR,
NO MORE THAN 7 CHARACTERS,DECIMAL POINT
MUST BE TYPED
DEPRESS JUMP KEY AFTER EACH COORDINATE
X(1)=-0.306
X(2)=-0.259
X(3)=-0.316
X(4)=0.860_
X(5)=
X(6)=
X(7)=
X(8)=
IF PREVIOUS VECTOR OK,JUST PRESS EOB
```

PRESS ONE KEY CORRESPONDING TO THE AXIS - VARIABLE YOU WISH TO SELECT, OR 30 IF YOU WISH TO STOP. IF YOU WISH TO MAKE CHANGES IN YOUR VECTOR PRESS KEY 28.

2-dimensional plane, the other axis being an observable variable. Notice that vectors which transform the original data points into overdetermined subspaces appear in this display for the user's ready reference. The user should try one or more of these vectors but he may also supply any number of other vectors, if he feels that this would help him interpret the structure of his data. The coordinates of the vector intended to be used should be typed in through the alphameric keyboard. The place where the digit (or any character for that matter) typed will appear on the screen is indicated by a cursor. The use of the "JUMP" key after one coordinate is entered, will cause the cursor to move over to the place for the next coordinate. After all the coordinates of a vector are entered, the user should press EOB. This is done by pressing both the 'ALT' and the '5' key of the alphameric keyboard. The first time the user is asked to supply a vector, he must give a non-null vector. (Later on, null vectors will be acceptable and simply mean that there is no change. At that time the user would just press EOB when this display appears and thus indicate that he does not wish to change the previous vector.)

After the vector is supplied, the user is asked to choose a variable as the other axis of a 2-dimensional plane. The choice is made by pressing the programmed function key corresponding to the number of the variable; if variable number 1 is desired, press key 1, etc.

After a vector and a variable have thus been chosen, the desired 2-dimensional plane will appear on the screen. The abscissa corresponds to the variable, the ordinate corresponds to the chosen vector. The legend (numbers given alongside the coordinate axes) indicates minimum and maximum values. The serial number of each data point is projected at the appropriate coordinates. Ellipses, i.e., projections of unit ellipsoids,

based upon the within-cluster metric of all points, are drawn around each cluster center. These ellipses generalize the concept of a standard unit interval in one dimension.

One aspect to be observed on the screen at this time is whether the points which supposedly belong to the same cluster, are close together (regardless which axes or vectors are used), and whether one could distinguish them visually from points belonging to a different cluster. There may be some overlaps but the clusters should be visually distinct. If certain points exhibit the above phenomenon in all displays, then they can be regarded as forming a cluster.

The user has now a choice. He may wish to superimpose one of the overdetermined subspaces, or he may wish to change the vector, or the variable, or both. To change the vector, he presses key 29, to change the number of the variable, he presses key 31, and to superimpose an overdetermined subspace, he presses the key corresponding to the number of that subspace.

Superimposition of an overdetermined subspace results in the display of projections of unit ellipsoids having metric based on only those points which belong to the overdetermined subspace. If the plane of projection selected is orthogonal to the overdetermined subspace, the projections of unit ellipsoids under reference will be flat and elongated. Further, the projections of the points lying on the subspace will make a narrow band (almost resembling a straight line). The plane of projection would be orthogonal to the overdetermined subspace, if the vector which transforms the original data points into this overdetermined subspace is supplied as the desired vector. As many different planes as there are

dimensions can be constructed by taking this normal vector against each of the variable axes.

The plane of projection, in a way, is the direction from which we look at the ellipsoids embedded in the p-dimensional space. The ellipsoids having metric based on the points belonging to an overdetermined subspace must necessarily be flat and elongated. However, they must be viewed from the proper direction. Otherwise, this elongation may not appear or, in some instances, they will appear as very small circles.

Once again, the user can press key 29 to supply a new vector, key 31 to change the number of the variable and the key corresponding to the number of an overdetermined subspace to superimpose the subspace. The program continues in this manner. It can be terminated at any time by pressing key 30.

It should be noted that the projections onto a plane formed by any pair of observable variables is a special case. If the user desires to have projections onto the plane formed by variables 1 and 2, say, all he has to do is supply (1.0, 0.0, . . . , 0.0) as the vector and press key 2. In fact, since the program does not necessarily require normalized vectors as input, any vector of the form (a, 0, 0, . . . , 0) where $a \neq 0$, results in the selection of variable 1 as one of the axes.

6.5 An Illustration

The programs described above were used in the analysis of artificial data. The data consisted of 45 points and 4 variables. These data were generated in the following manner. First, 180 normal deviates with zero mean and unit variance were generated; they made up the 180 elements of a 45 x 4 matrix, numbered column-wise. The first 25 measurements on the 4th

variable were then redefined by the relation

$$z(I) = z(I)/10 + (z(I - 135) + z(I - 90) + z(I - 45))/3$$

$$I = 136, 137, \ldots, 160$$

i.e., the first 25 observations on variable 4 were replaced by one tenth of the original observation plus the mean of the first three variables. Similarly, the last 25 observations on variable 3 were replaced by the relation

$$z(I) = z(I)/10 + (z(I - 90) + z(I - 45) + z(I - 45))/3$$

$$I = 111, 112, \ldots, 135$$

The modification of the data matrix by these two operations built in 2 subspaces. In effect, the first 25 observations on variable 4 became almost a linear combination of the remaining 3 variables and the last 25 observations on variable 3 similarly became almost a linear combination of the remaining 3 variables. Still, however, all the variables had zero mean, and to introduce mean shifts, the vectors (3, 9, 5, 9), (5, 9, 7, 3) and (7, 3, 7, 5) were added to the first 15 observations, second 15 observations and the last 15 observations respectively. Thus the entire data matrix became a simulated sample drawn from normal populations with mean vectors (3, 9, 5, 9), (5, 9, 7, 3) and (7, 3, 7, 5) and two built in subspaces. This data matrix was then subjected to the first program of cluster identification and subspace determination. This program correctly assigned the first 15 points to one cluster, the second 15 points to another cluster and the last 15 points to a third cluster. The subspace identification part of the program gave 3 overdetermined subspaces. This was not at all surprising since the two planes were already built in and, as frequently happens in such instances (see section 4.2), a plane was found somewhat in-between the two constructed planes. The cosines between

the normals to these three planes were as follows:

	1	2	3
1	1.00000	-0.29684	0.00673
2	-0.29684	1.00000	0.02303
3	0.00673	0.02303	1.00000

The planes identified were as under (serial numbers of points belonging to the planes are given).

Plane 1: --12, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, (24 points)

Plane 2: --2, 9, 14, 15, 17, 18, 29, 30, 32, 34, 37, 39, (12 points)

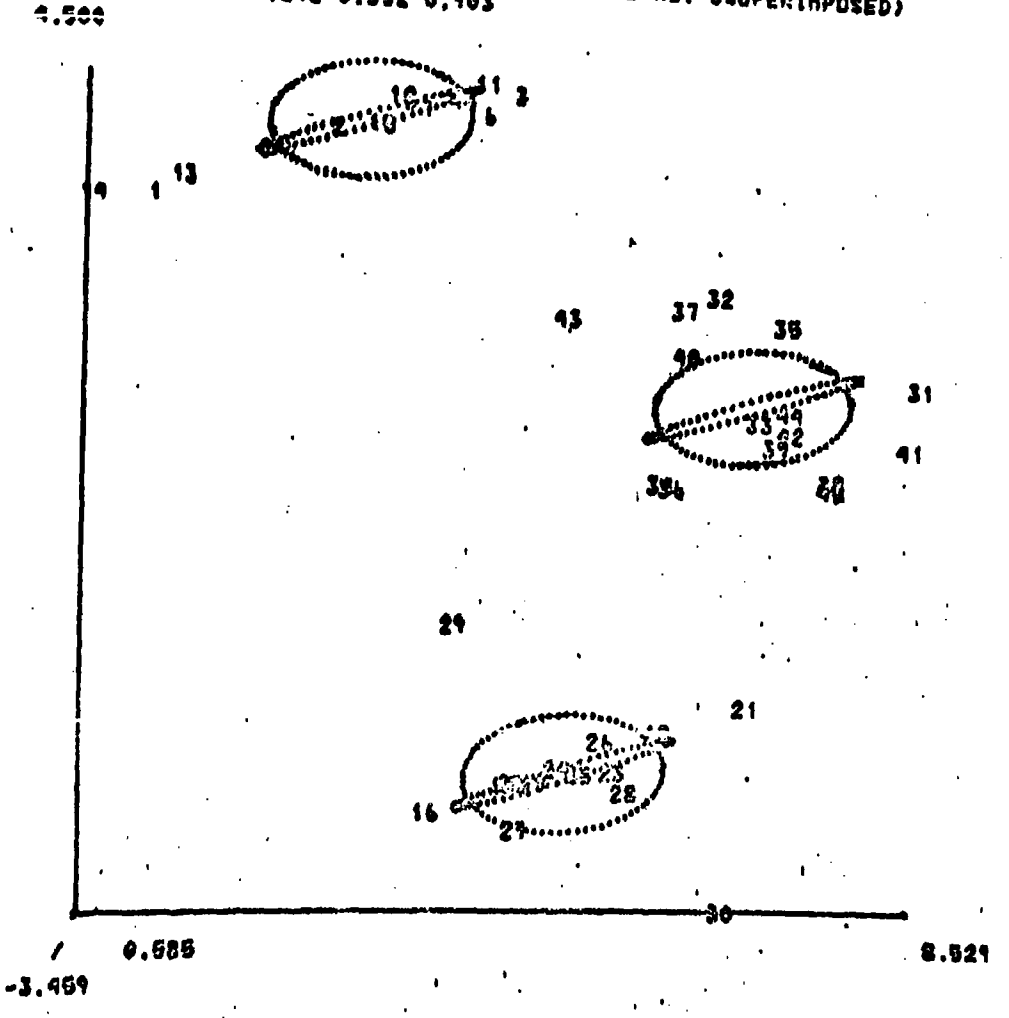
Plane 3: --1, 2, 4, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, (20 points)

According to the built-in subspaces, one subspace should have contained the last 25 points, i.e., points 21-45 and the other subspace should have contained the first 25 points. Plane 1 given above did pick up 23 of the last 25 points and an extra point number 12. Likewise, plane 3 picked up 20 of the first 25 points. Plane 2 picked up 6 of the points belonging to plane 1 and 6 of the points belonging to plane 3. Thus plane 2 is somewhat of a mixture of the planes 1 and 3.

The results of the CLUSTR program were then displayed using the ELLIPSE program. Notice especially the following displays in which the 2-dimensional planes were formed by selecting a suggested vector (-0.306, -0.259, -0.316, 0.860) and an observable variable. The vector under consideration is the vector 3, which transformed the original data points into simple structure plane 3.

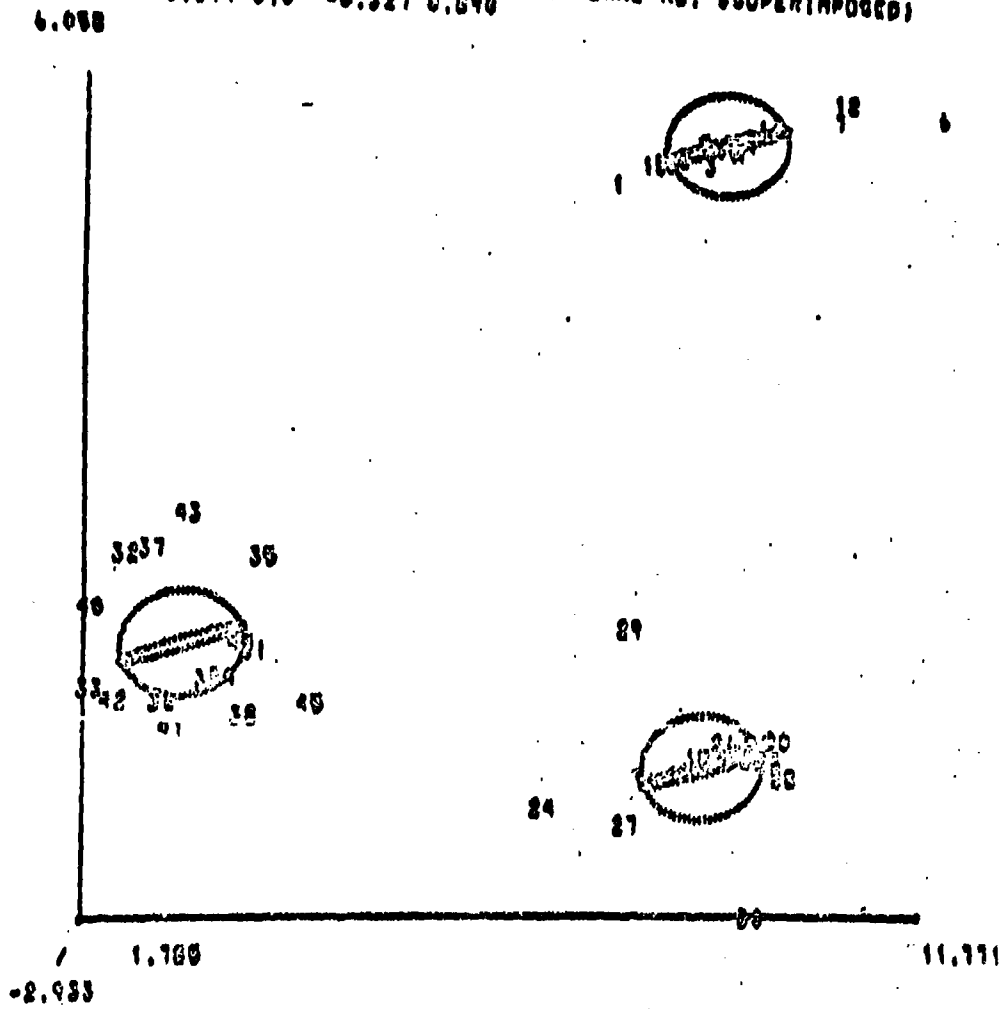
(a) Variable 1 against vector 3--After normalizing, the vector reduced to (0.0, -0.272, -0.332, 0.903). Plane 3 was superimposed. Points number 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 can be seen to be lying on the simple structure plane.

VARIABLE 1AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 3SUPERIMPOSED)
0.0 -0.272-0.332 0.403



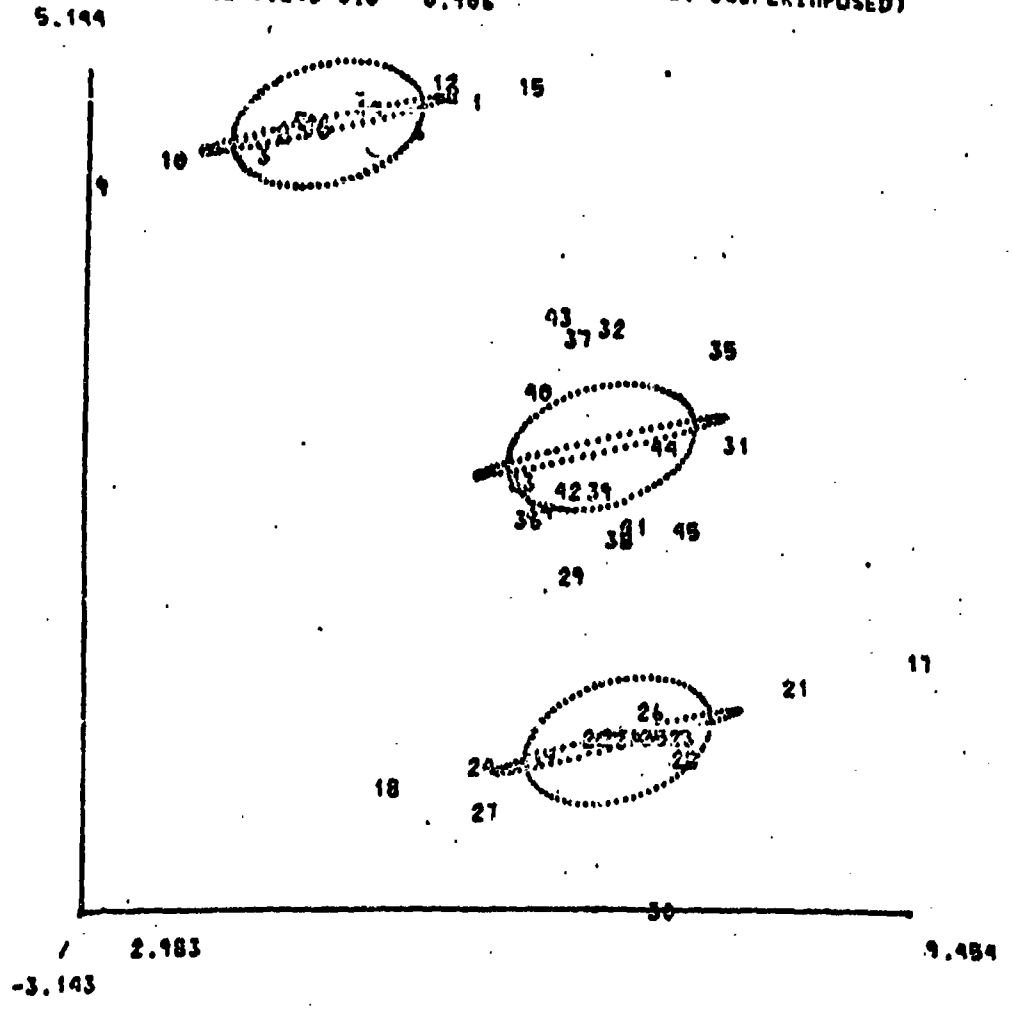
(b) Variable 2 against vector 3--After normalizing, the vector reduced to (0.317, 0.0, 0.327, 0.000). Plane 3 was superimposed. Points number 1, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 can be seen lying on the simple structure plane.

VARIABLE 24AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 3SUPERIMPOSED)



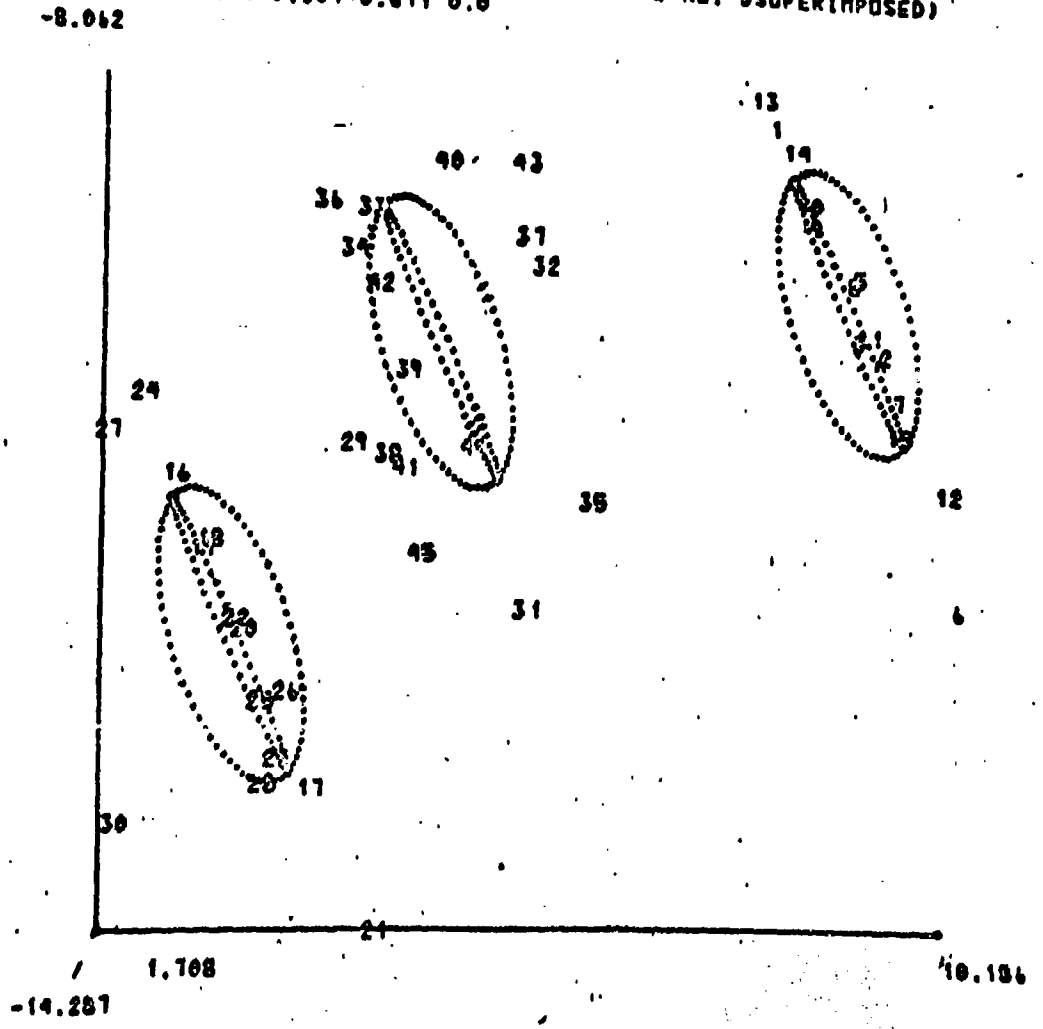
(c) Variable 3 against vector 3--After normalizing, the vector reduced to (-0.322, -0.273, 0.0, 0.906). Plane 3 was superimposed. Points number 1-26 can be seen lying on the simple structure plane.

VARIABLE 3AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 3SUPERIMPOSED)
-0.322-0.273 0.0 0.906



(d) Variable 4 against vector 3--The vector, after normalizing, reduced to (-0.599), -0.507, -0.619, 0.0). Plane 3 was superimposed. Points number 1-27 with the exception of number 5 can be seen lying on the simple structure plane.

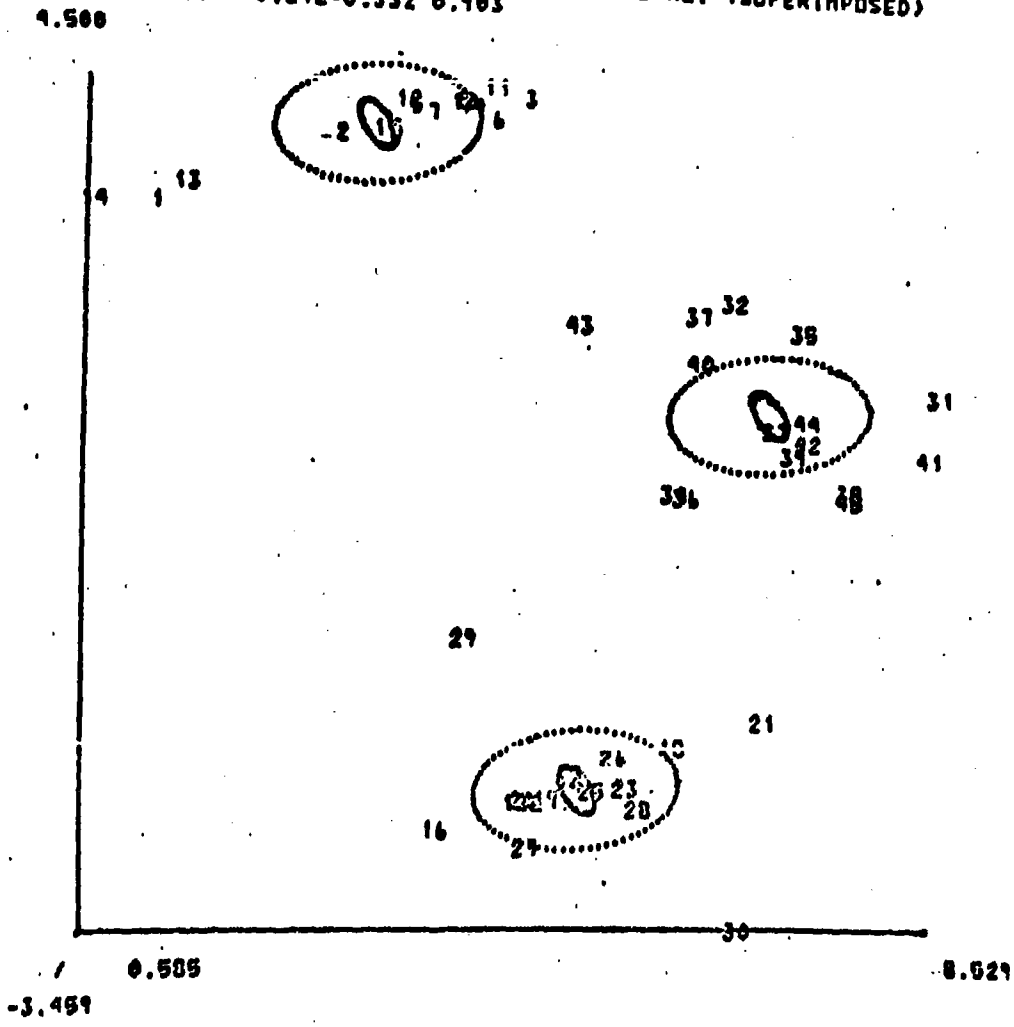
VARIABLE 9AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 3SUPERIMPOSED)
-0.549-0.507-0.619 0.0



The above 4 displays pertain to each of the 4 variables against vector 3 and superimposition of simple structure plane 3. Vector 3 is the vector which transforms the raw data into simple structure plane 3. Points 1-25 with the exception of 3, 5, 6, 12 and 23 lie on this plane. Vector 3 is the normal to this simple structure plane 3. Hence any plane passing through vector 3 is orthogonal to the simple structure plane 3. When projections onto this orthogonal plane are taken, the points lying on the simple structure plane should fall within a narrow band (almost resembling a straight line) and the above 4 displays clearly bring out this fact. Variables 1, 2, 3, and 4 make 4 different planes, respectively, passing through vector 3 and all orthogonal to the simple structure plane 3. This can also be thought of as rotating a plane passing through vector 3 around the vector 3. Projections are taken when this rotating plane passes through the axis corresponding to each of the variables. Attention should also be drawn to these displays.

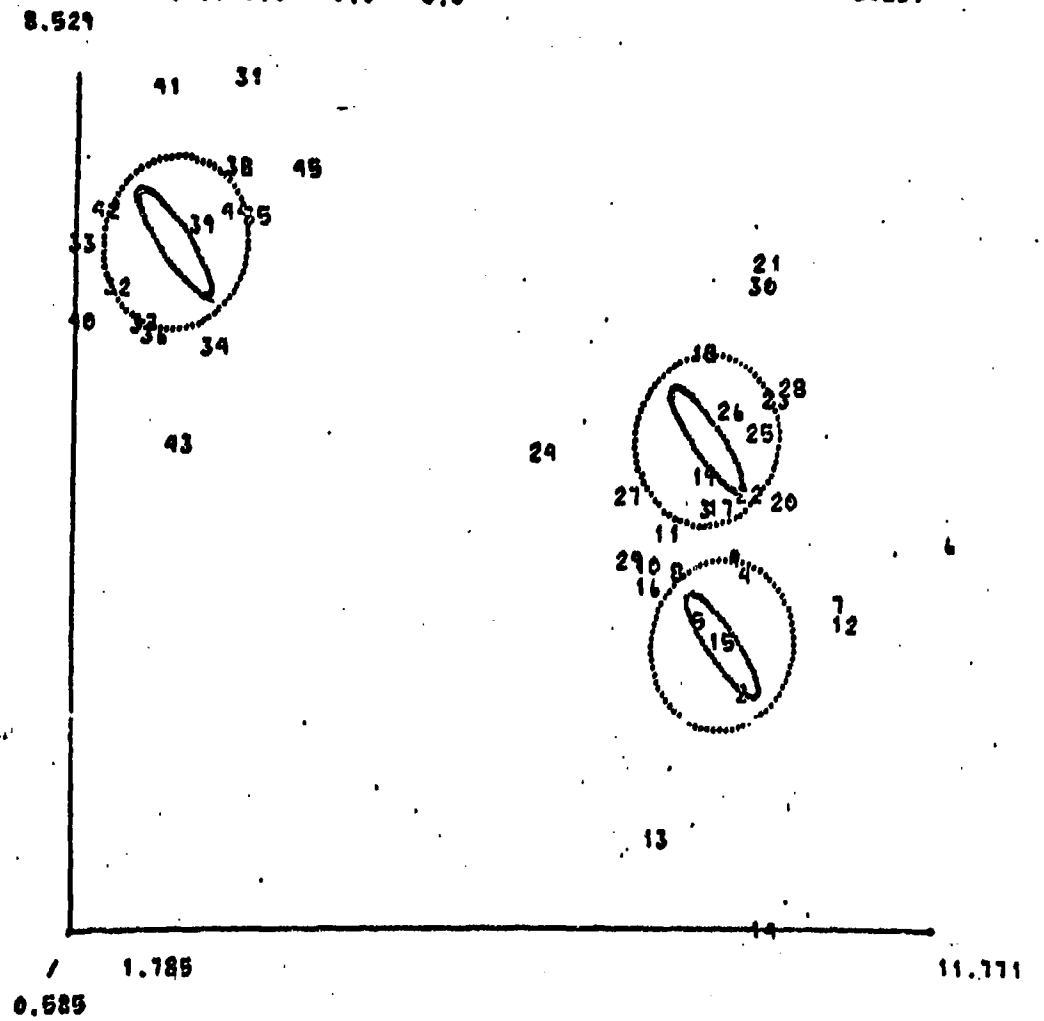
(e) Variable 1 against vector 3--The vector, after normalizing, reduced to (0.0, -0.272, -0.332, 0.903). Simple structure plane 1 was superimposed. Since the plane passing through vector 3 and the axis corresponding to variable 1 is not orthogonal to simple structure plane 1, we do not see points lying on a narrow band resembling a straight line in this display.

VARIABLE 1AGAINST VECTOR BELOW,(SIMPLE PLANE NO. 1SUPERIMPOSED)
0.0 -0.272-0.332 0.903



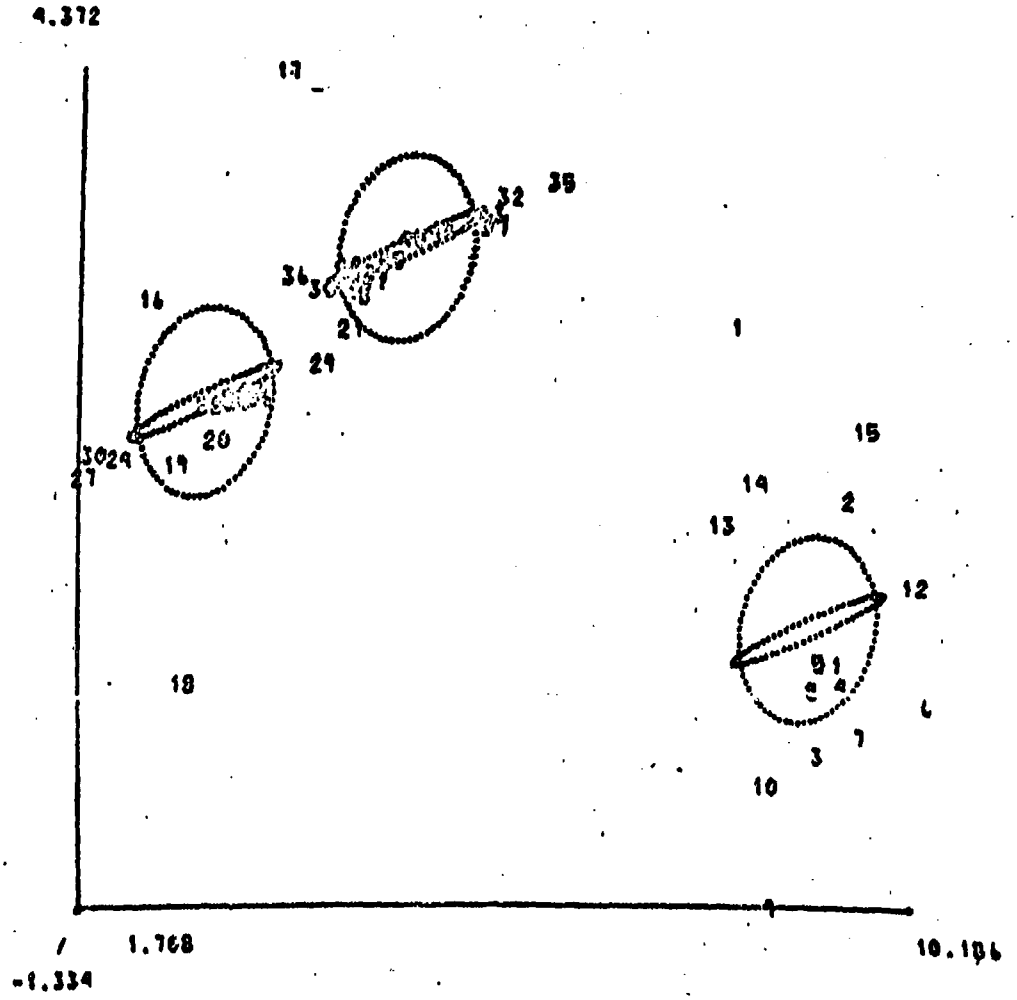
(f) Variable 2 against variable 1--Plane 3 was superimposed. Once again, for the reasons mentioned in (e) above, we do not see a good simple structure in this display. We simply are not looking at the ellipsoids from the proper position.

VARIABLE 2AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 3SUPERIMPOSED)
1.000 0.0 0.0 0.0



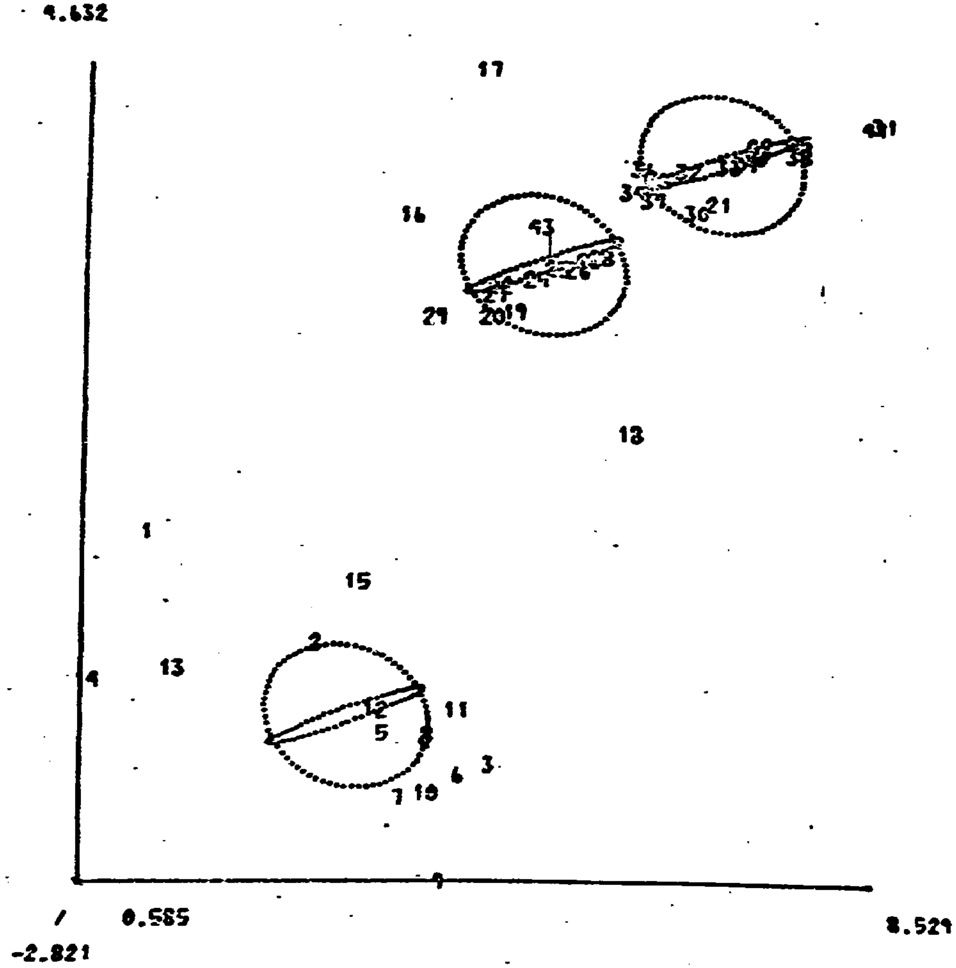
(g) Variable 4 against vector 1--The vector, after normalizing, reduced to (-0.325, -0.289, 0.900, 0.0). Plane 1 was superimposed. Since any plane passing through vector 1 is orthogonal to simple structure plane 1, we expect to see a good simple structure in this display and we do. Points number 21, 22, 23, 25, 26, 28, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, lie within a narrow band resembling a straight line.

VARIABLE AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 1SUPERIMPOSED)
-0.525-0.289 0.900 0.0

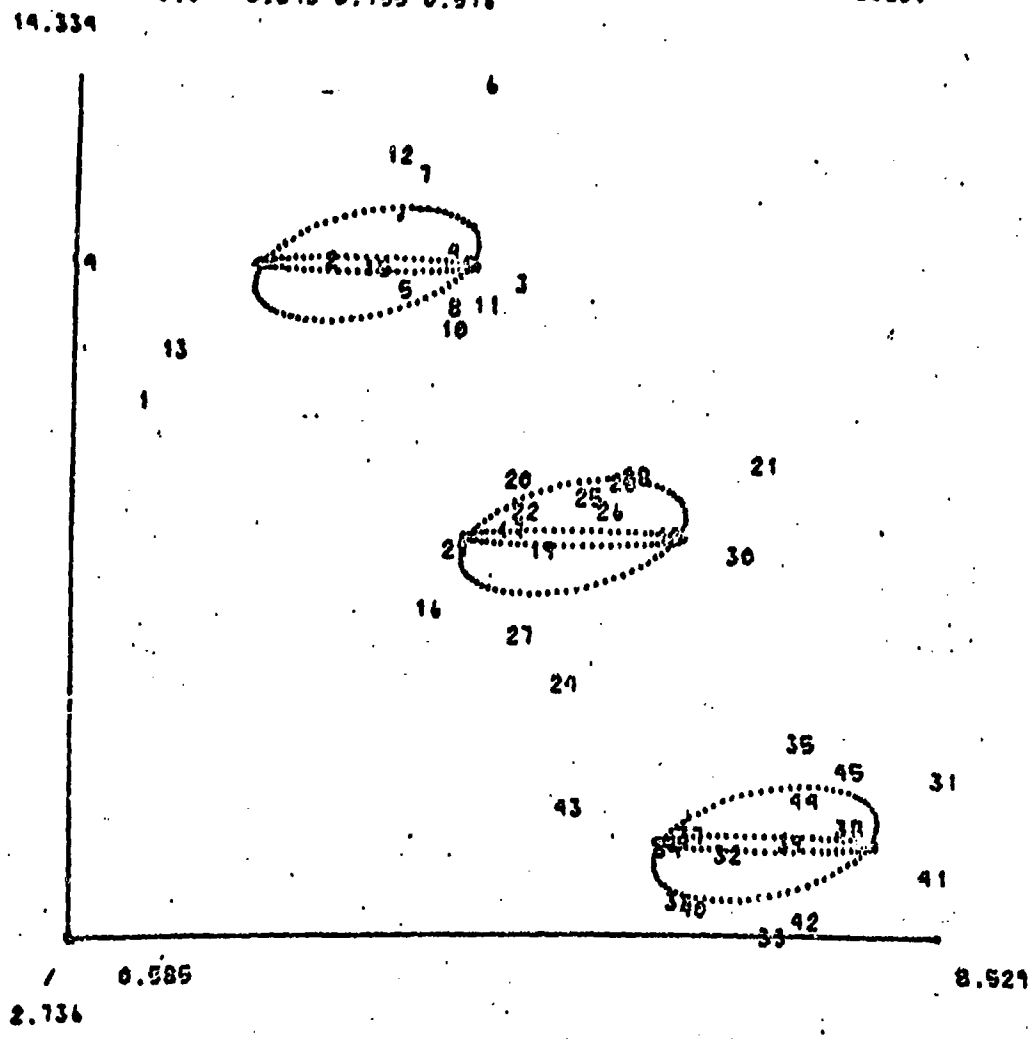


The user should find the other displays easy to understand. A good simple structure is seen only when a plane is selected which passes through a vector that transforms the raw data into a simple structure solution, and the corresponding simple structure plane is superimposed. It should, however, be noted that the clustering of points is not affected by this principle and hence, in all displays, the clusters can be easily identified.

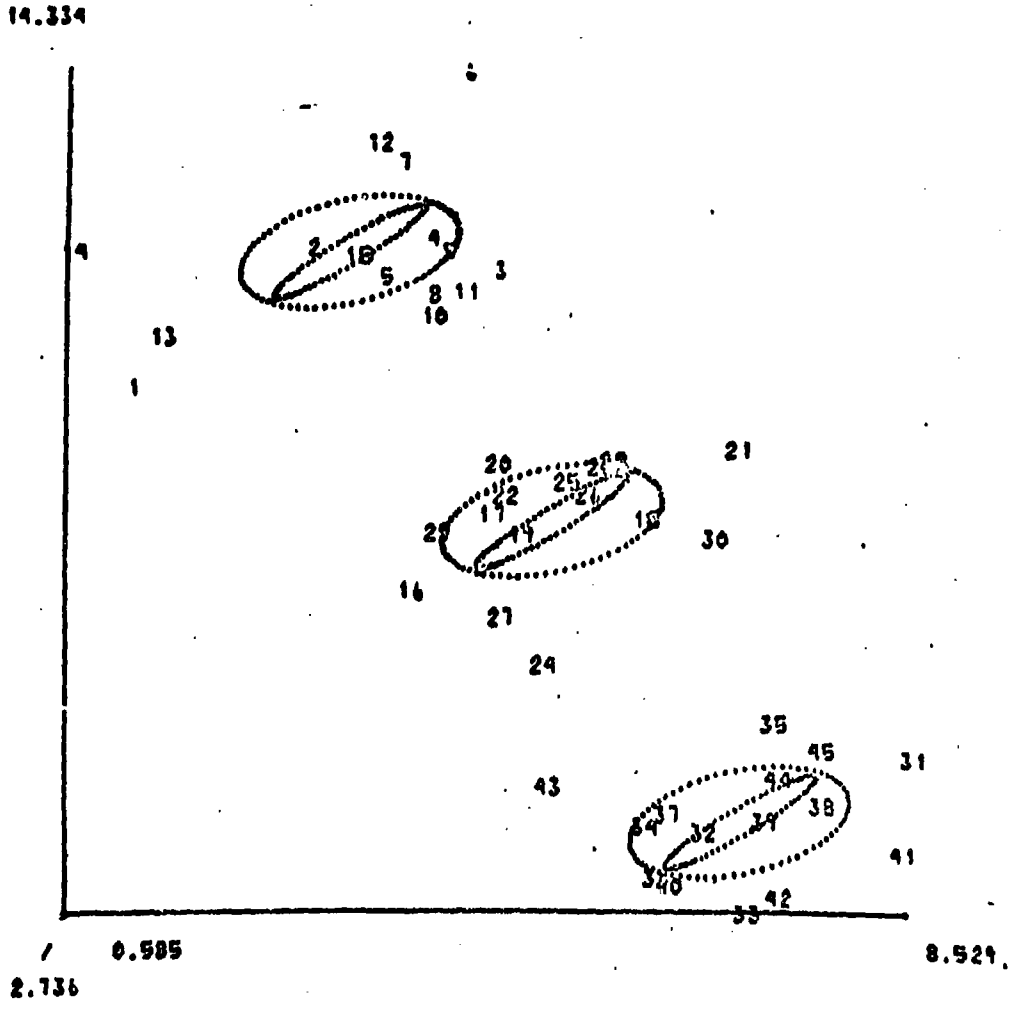
VARIABLE 1AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 1SUPERIMPOSED)
0.6 -0.289 0.900-0.325



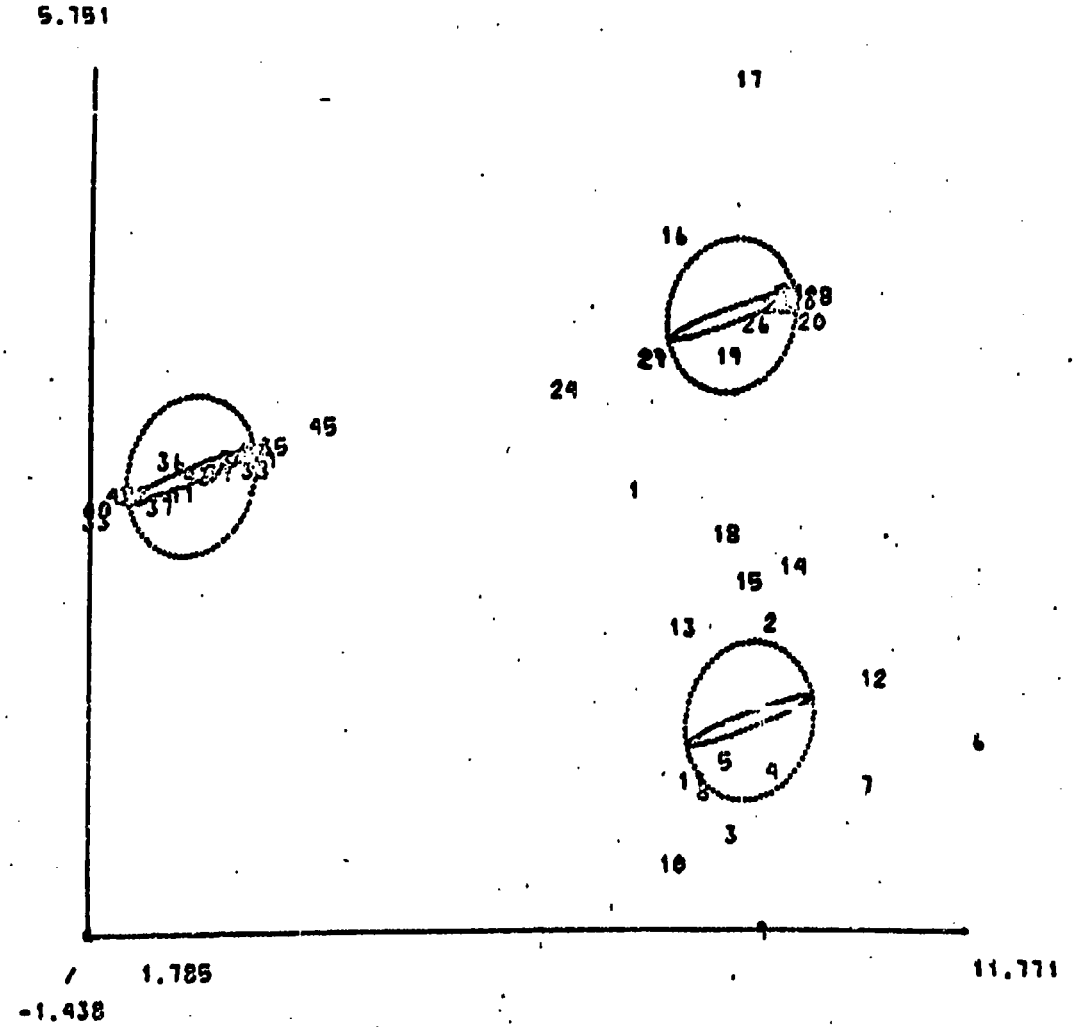
VARIABLE 1AGAINST VECTOR BELOW.(SIMPLE PLANE NO. 2SUPERIMPOSED)
0.0 0.893-0.155 0.516



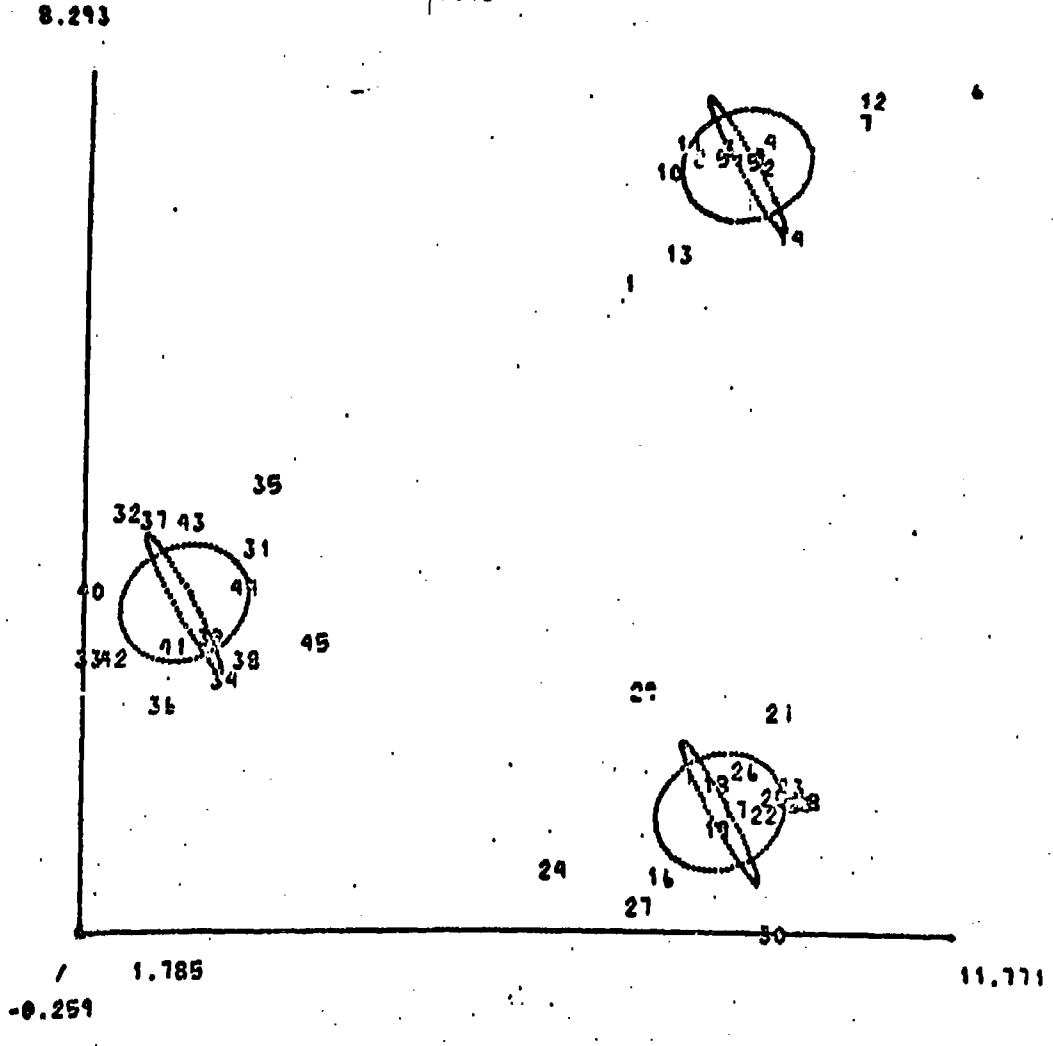
VARIABLE 1AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 3SUPERIMPOSED)
0.0 0.843-0.155 0.516



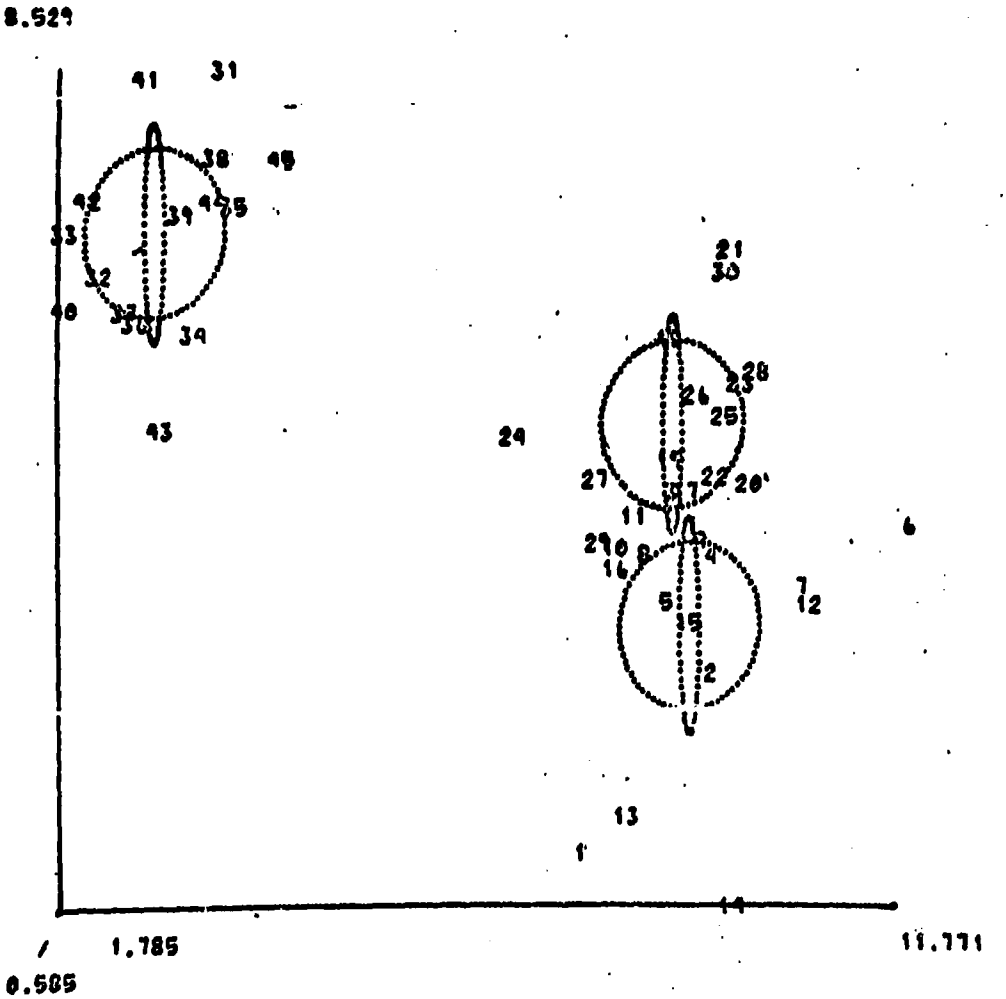
VARIABLE 2AGAINST VECTOR BELOW.(SIMPLE PLANE NO. 1SUPERIMPOSED)
-0.322 0.0 0.891-0.322



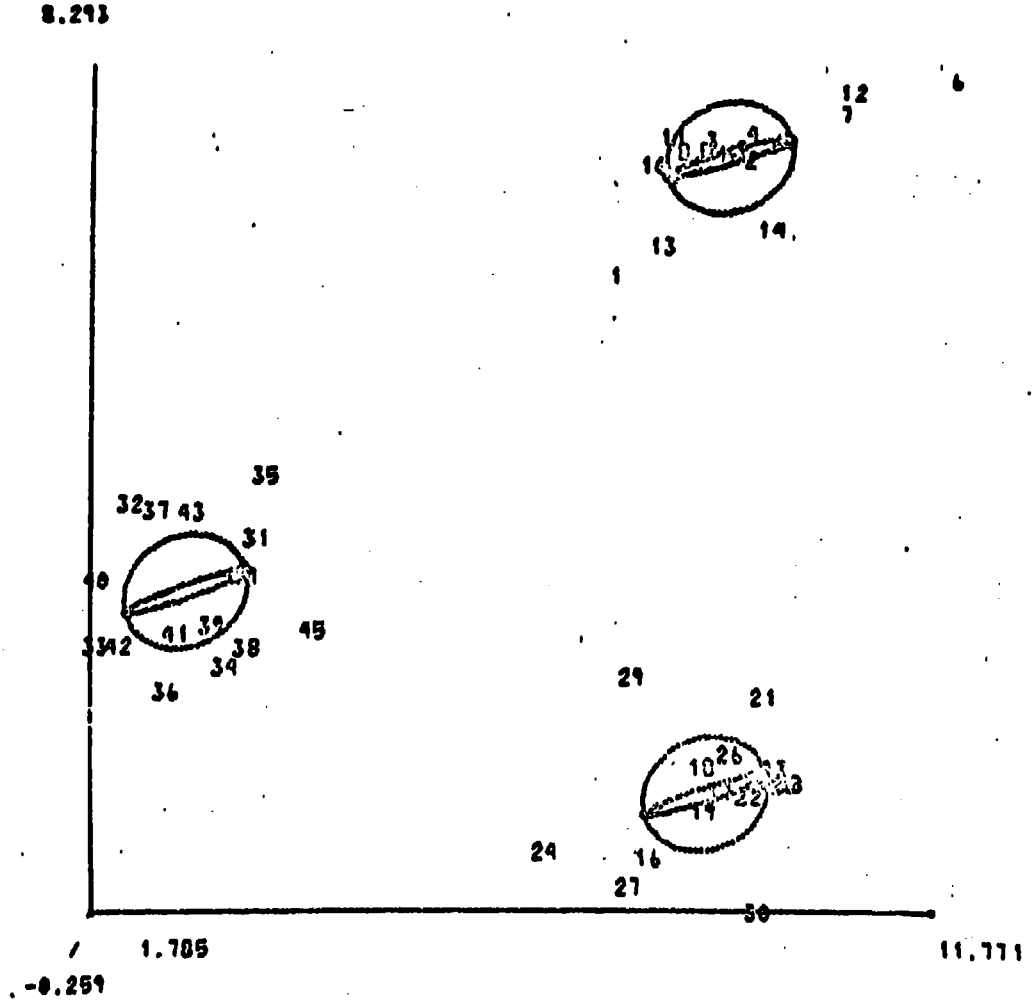
VARIABLE 2AGAINST VECTOR BELOW,(SIMPLE PLANE NO. 2SUPERIMPOSED)
0.022 0.0 -0.208 0.951



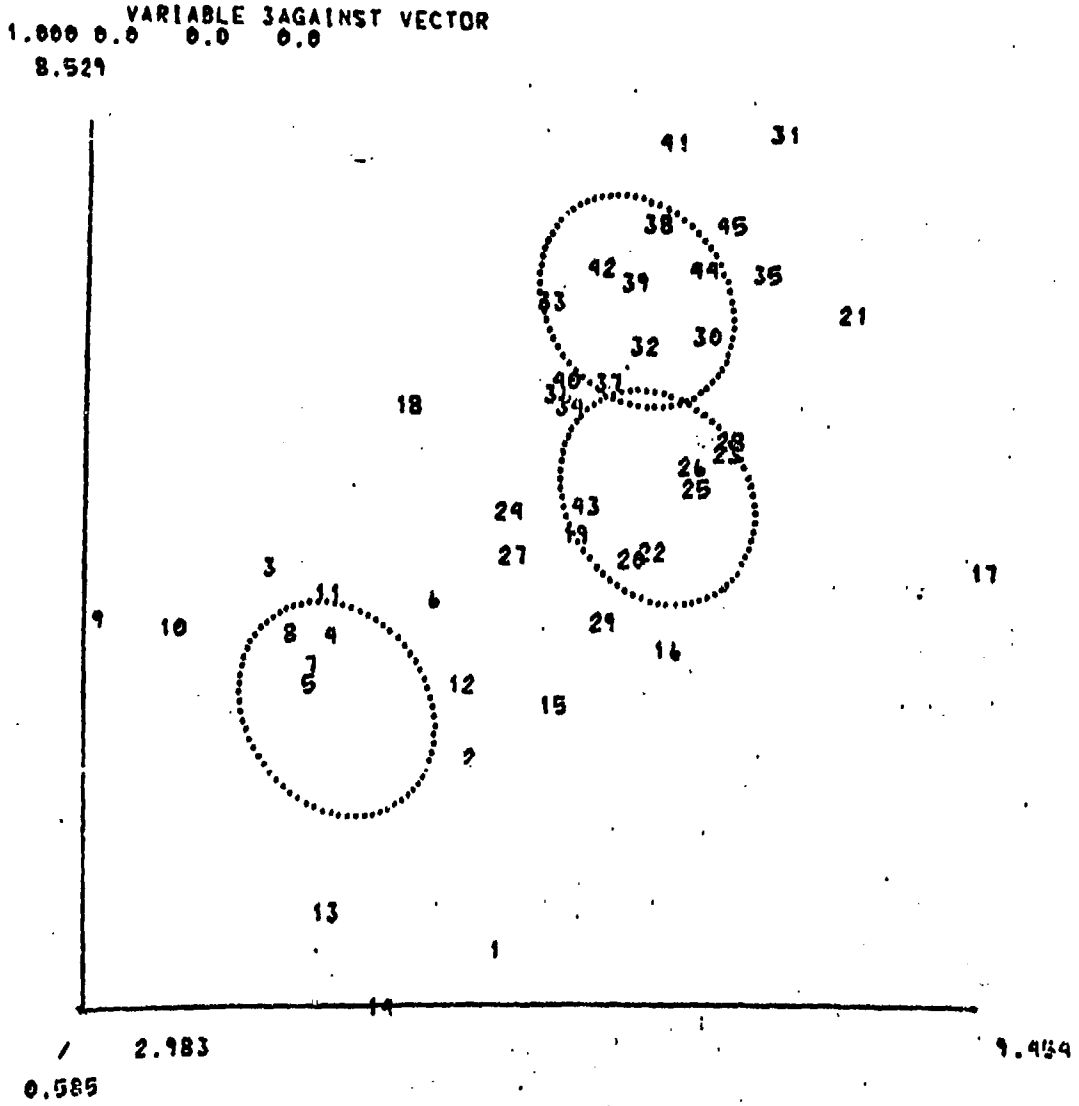
VARIABLE 2AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 2SUPERIMPOSED)
1.000 0.0 0.0 0.0



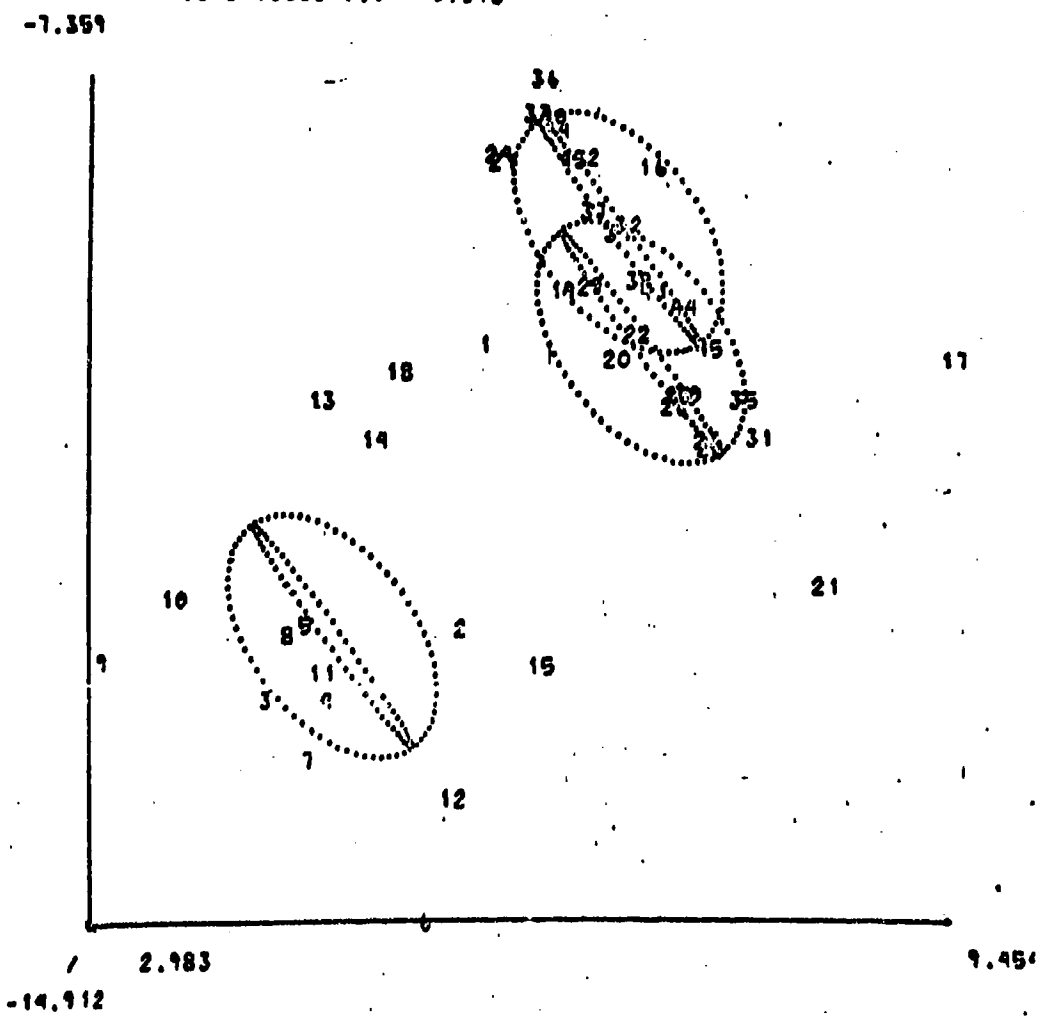
VARIABLE 2AGAINST VECTOR BELOW.(SIMPLE PLANE NO. ;SUPERIMPOSED)
0.022 0.0 -0.260 0.957



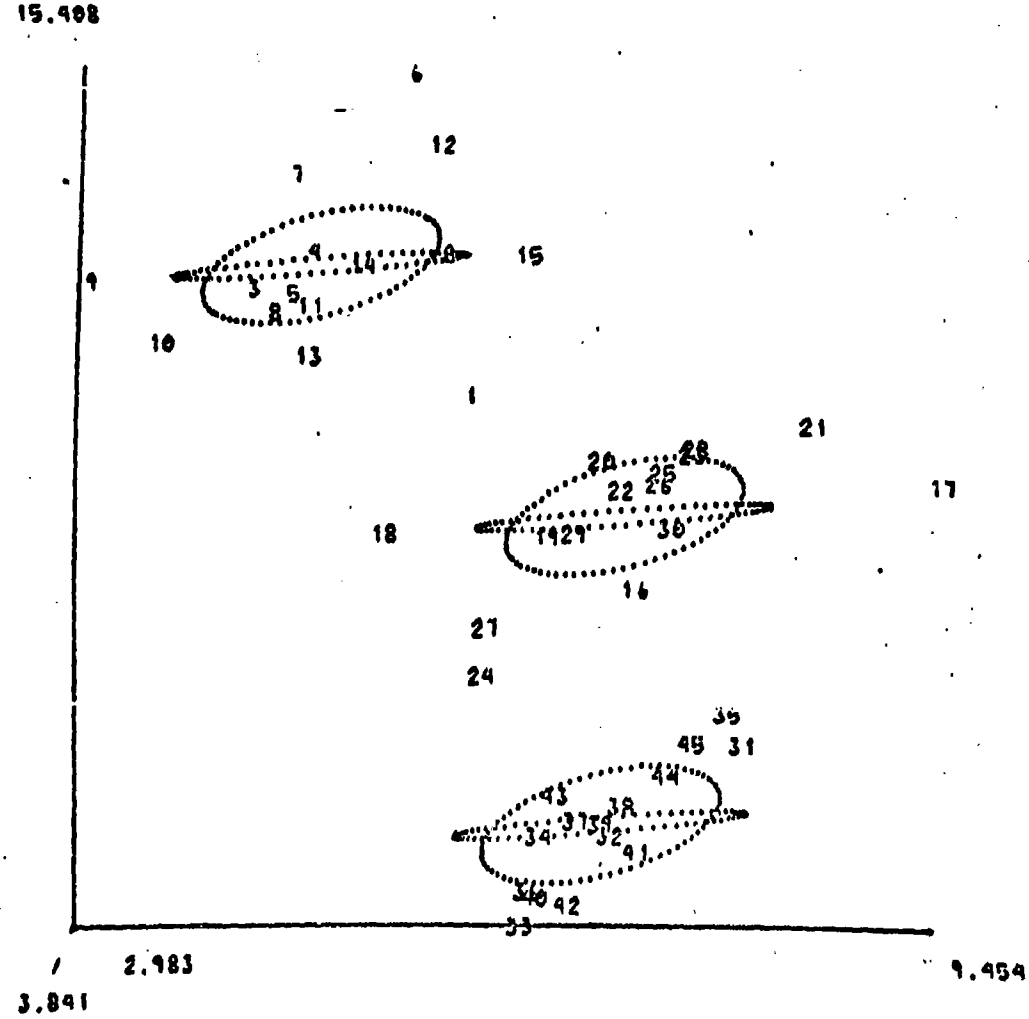
VARIABLE 3AGAINST VECTOR



VARIABLE 3AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 1SUPERIMPOSED)
-0.598-0.533 0.0 -0.598

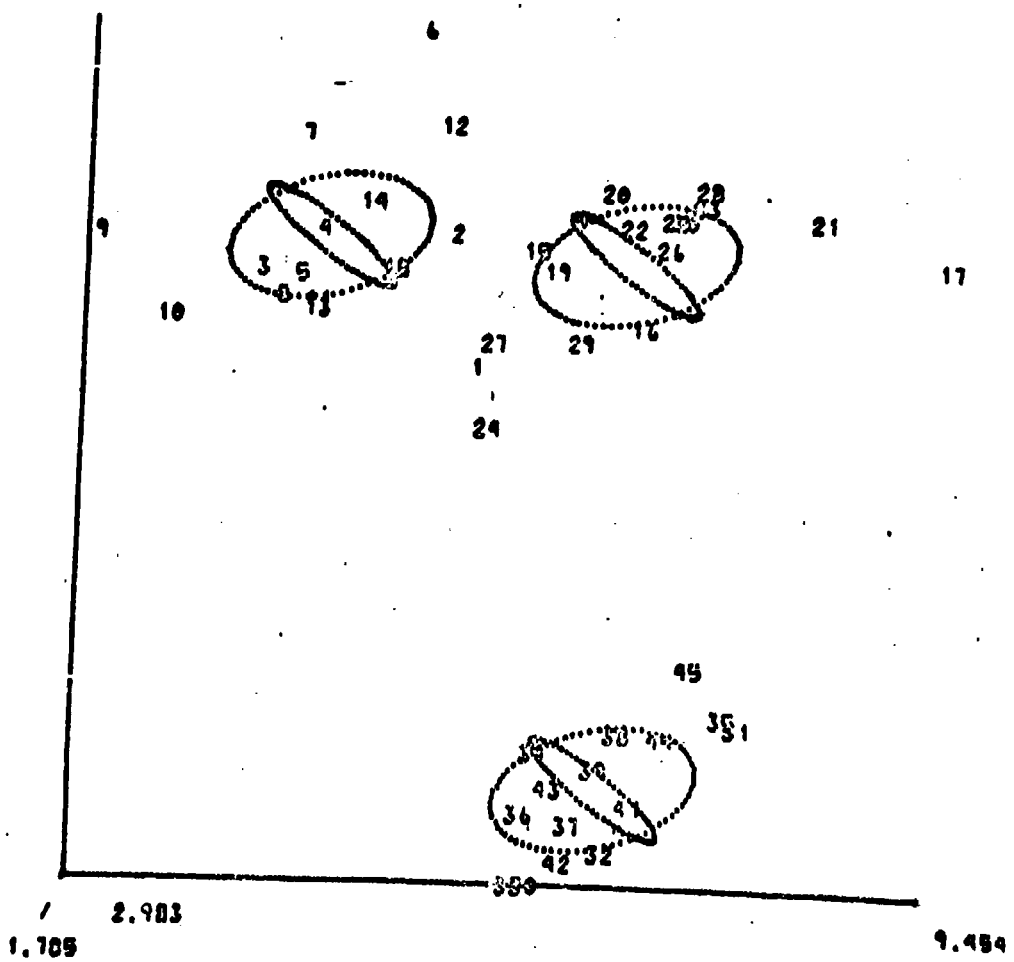


VARIABLE 3AGAINST VECTOR BELOW.(SIMPLE PLANE NO. 2SUPERIMPOSED)
0.012 0.853 0.0 0.522

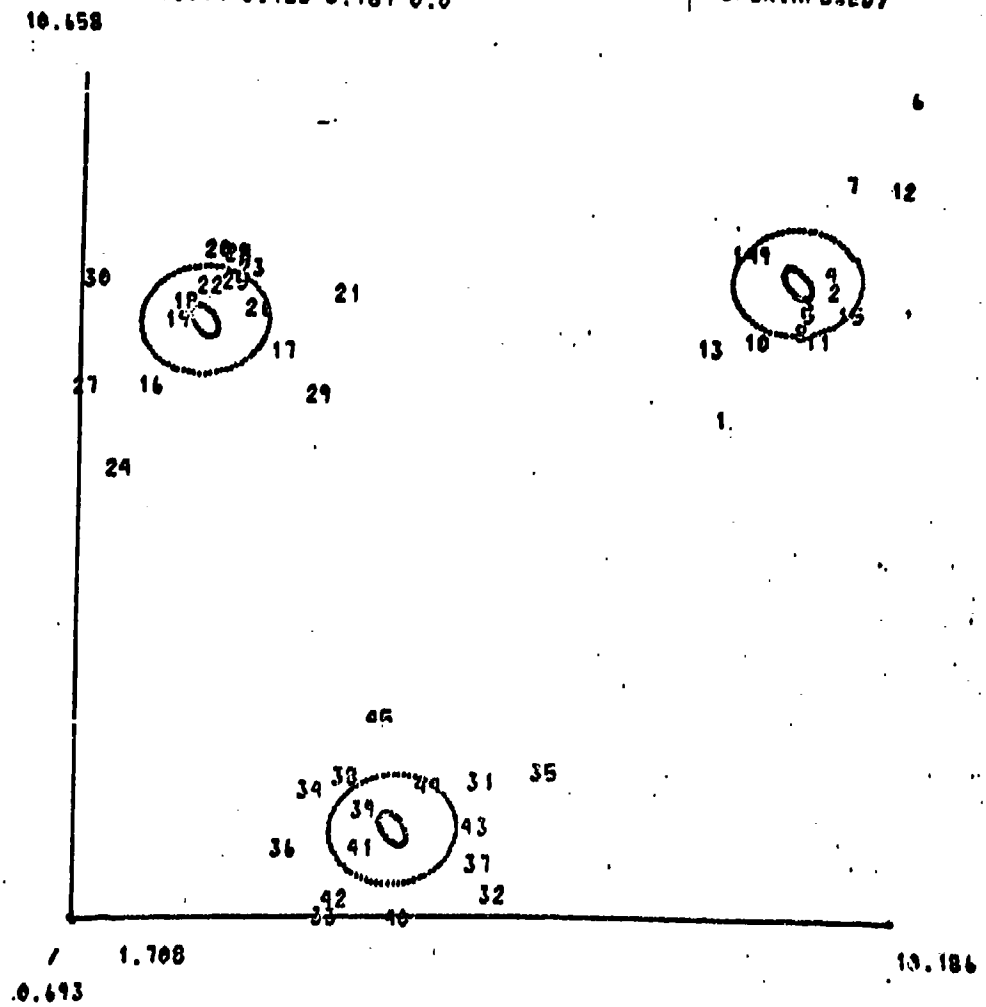


VARIABLE 3AGAINST VECTOR BELOW;(SIMPLE PLANE NO. 3SUPERIMPOSED)
0.0 1.000 0.0 0.0

11.771



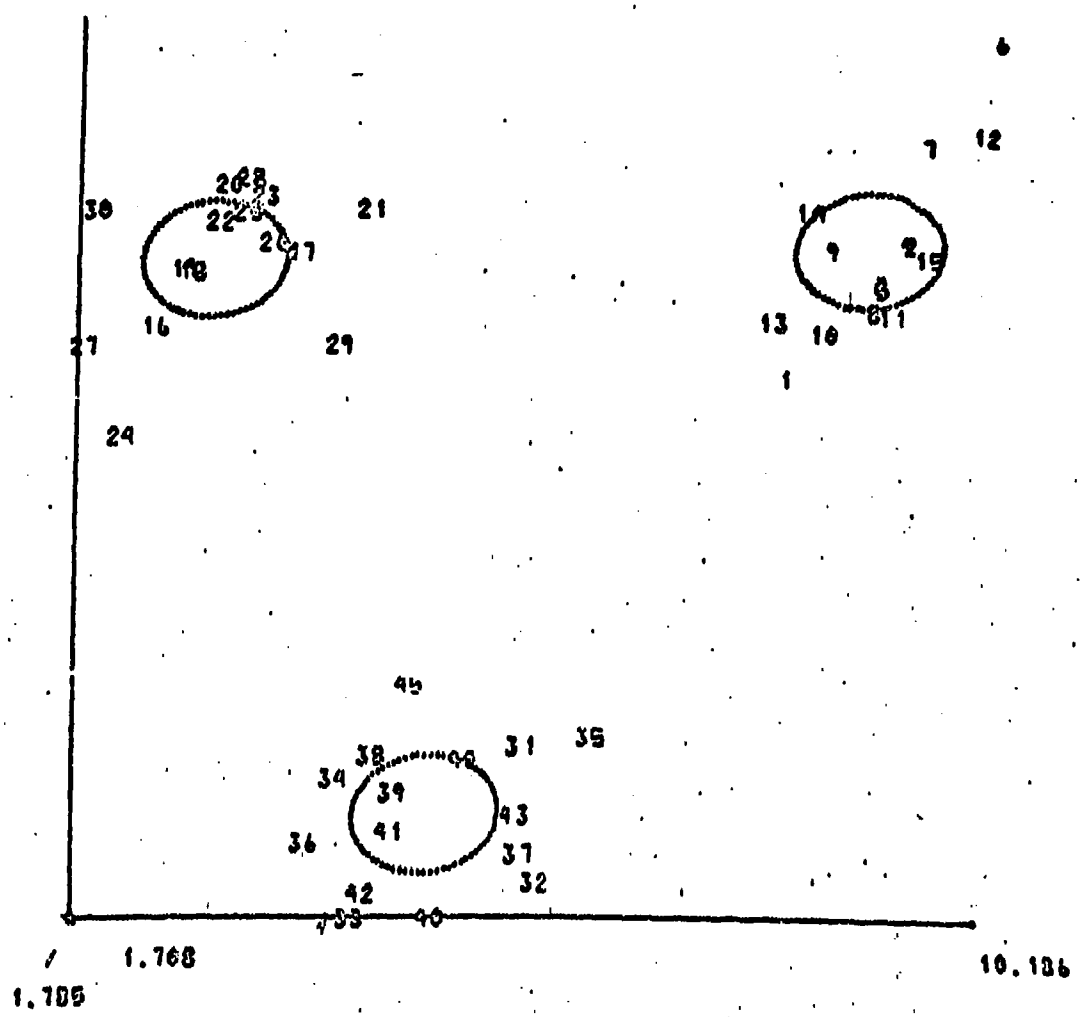
VARIABLE 9AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 2SUPERIMPOSED)
0.019 0.183-0.181 0.0



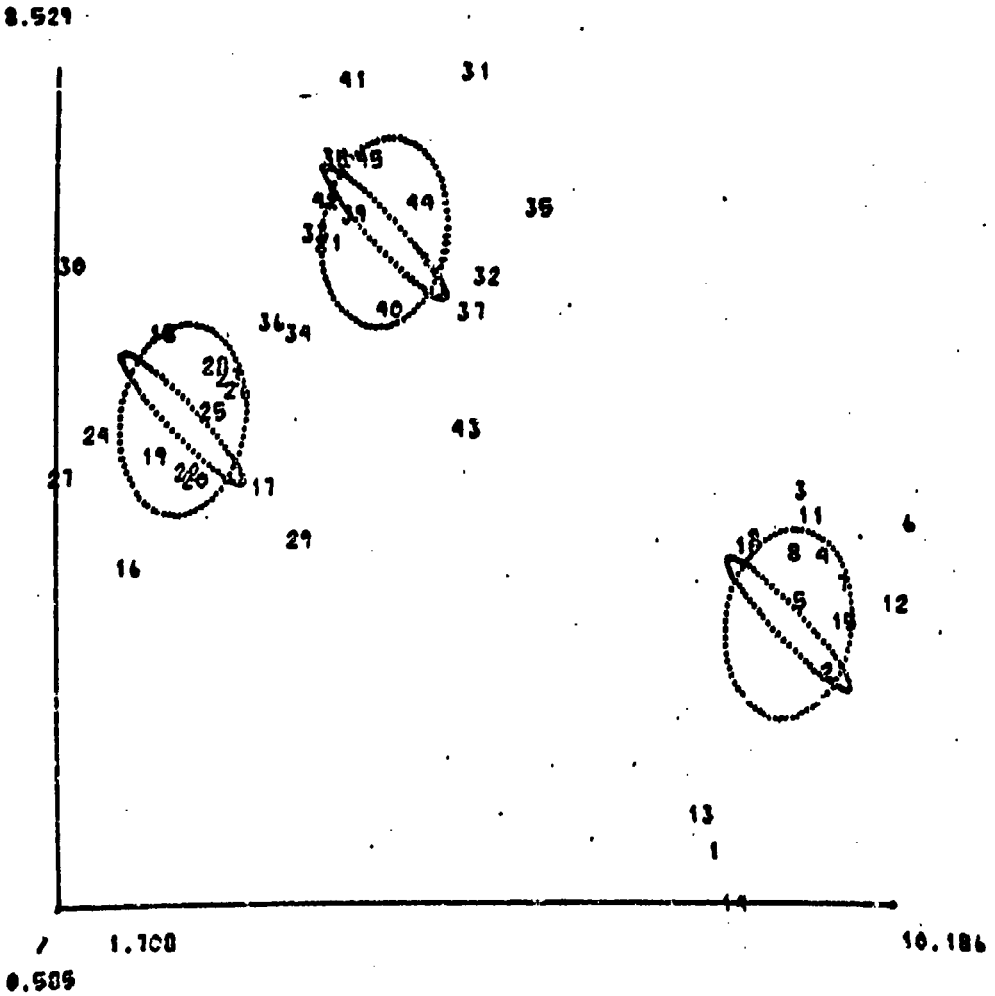
VARIABLE AGAINST VECTOR

0.0 1.000 0.0 0.0

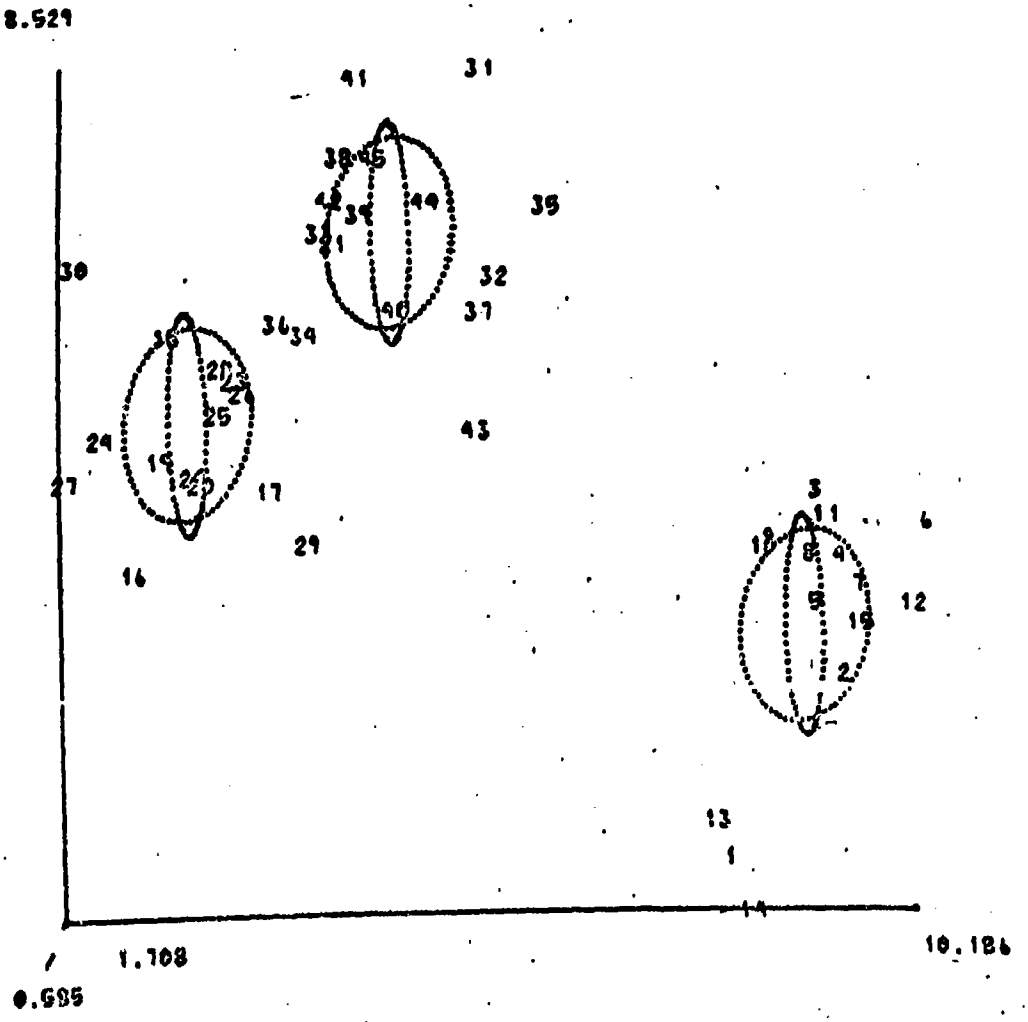
11.771



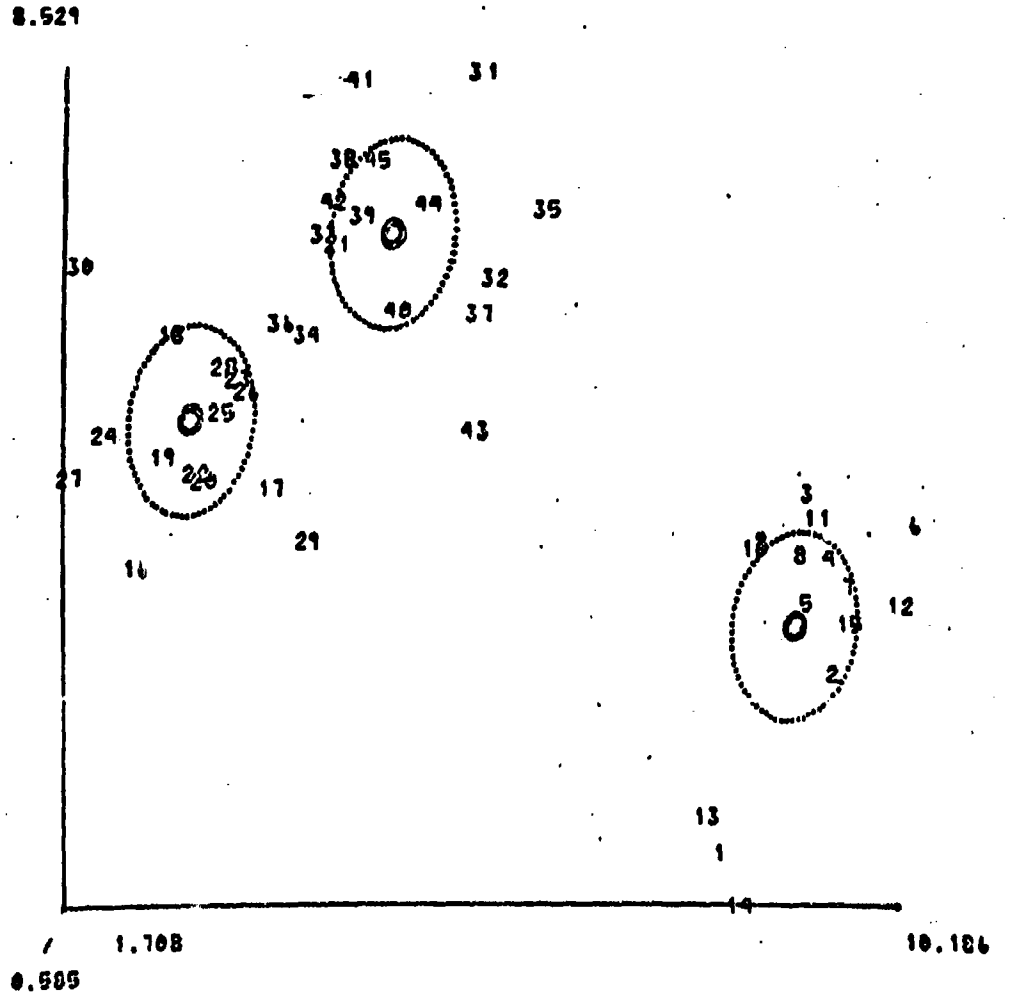
VARIABLE 4AGAINST VECTOR BELOW,(SIMPLE PLANE NO. 1SUPERIMPOSED)
1.000 0.0 0.0 0.0



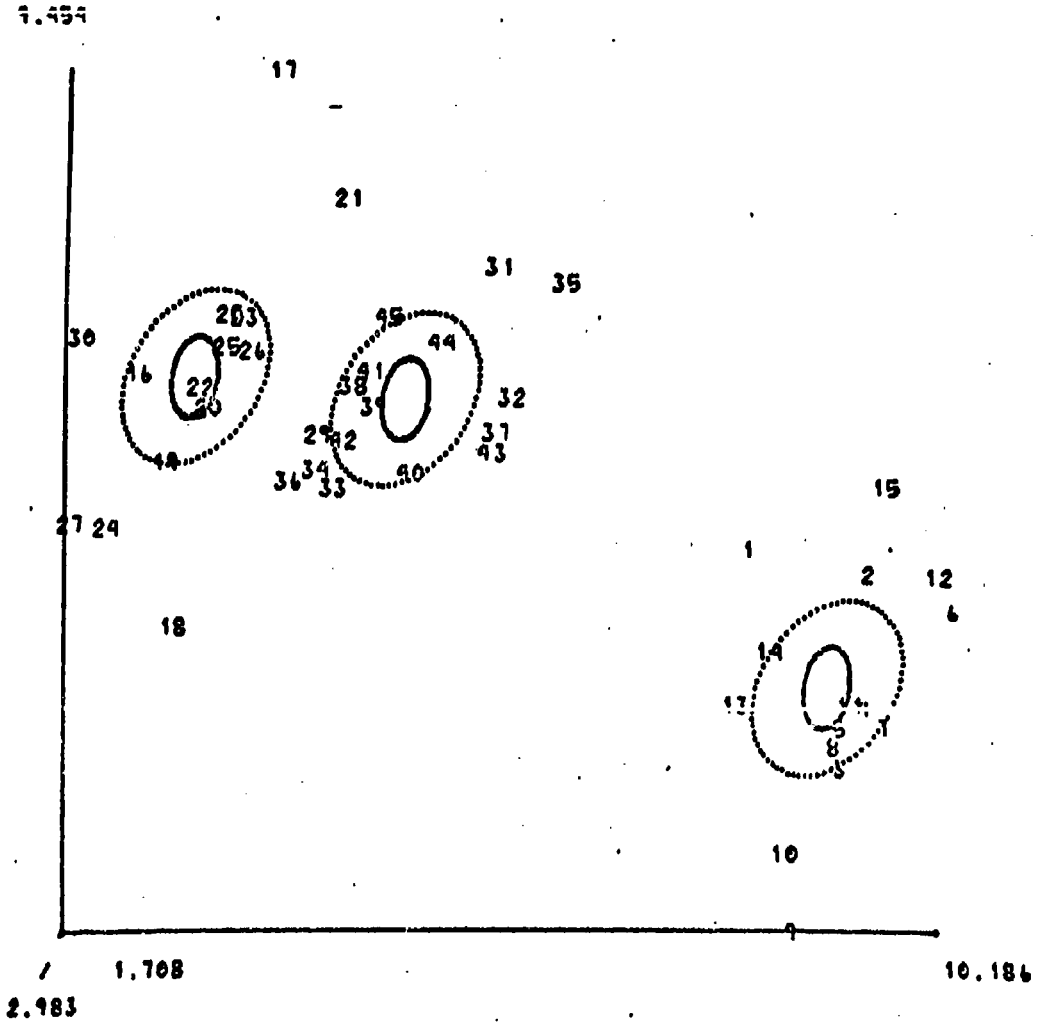
VARIABLE 4AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 2SUPERIMPOSED)
1.000 0.0 0.0 0.0



VARIABLE 4AGAINST VECTOR BELOW,(SIMPLE PLANE NO. 3SUPERIMPOSED)
1.000 0.0 0.0 0.0



VARIABLE 9AGAINST VECTOR BELOW:(SIMPLE PLANE NO. 2SUPERIMPOSED)
0.0 0.0 1.000 0.0



BIBLIOGRAPHY

(1) Abramowitz, M., and Stegun, I., Handbook of Mathematical Functions, National Bureau of Standards, Washington, D. C., 1964.

(2) Bargmann, R., "Signifikanzuntersuchungen der Einfachen Struktur in der Faktoren-Analyse," Mitteilungsblatt Fur Mathematische Statistik, 1954.

(3) Bargmann, R., "Factor Analysis," unpublished address presented at the 2nd annual Goddard Computer Science Symposium, Dallas, 1966.

(4) Bargmann, R., and Richard Graney, Tables for Significance Tests for Virtual Clusters, Technical Report No. 36, University of Georgia, Dept. of Statistics.

(5) Bargmann, R., and Richard Graney, An Algorithm for Identifying and Testing Virtual Clusters, Technical Report No. 42, University of Georgia, Department of Statistics.

(6) Cattell, R. B., Factor Analysis, Harper & Bros., New York, 1952.

(7) Edwards, A. W. F. and Cavalli-Sforza, L. L., "A method for Cluster Analysis," Biometrics, 21, 1965.

(8) Feller, W., An Introduction to Probability Theory and its Applications, No.s I and II, John Wiley & Sons, New York, 1966.

(9) Fortier, J. and Solomon, H., "Clustering Procedures," Multivariate Analysis, edited by P. R. Krishnaiah, Academic Press, New York, 1966.

(10) Graney, Richard W., Tests of Concentration and Identification of Mixed Samples, unpublished doctoral dissertation submitted to University of Georgia, 1969.

(11) Guttman, L. "A general nonmetric technique for finding the smallest coordinate space for a configuration of points," Psychometrika, vol. 33, pp. 469-506.

(12) Gower, J. C., "A comparison of some methods of cluster analysis," Biometrics, 1967.

(13) Gower, J. C., "Some distance properties of latent roots and vector methods used in multivariate analysis, Biometrika, 1966.

(14) Harman, H. H., Modern Factor Analysis, University of Chicago Press, Chicago, 1967.

(15) Holzinger, K. and Harman, H. H., Factor Analysis, University of Chicago Press, Chicago, 1941.

(16) Hurwitz, A., Yeaton, J. and Schaffer, R., Graphics Additions to FORTRAN (GRAF) 1968.

(17) IBM, System/360 Scientific Subroutine Package, White Plains, N. Y., 1968.

(18) Kendall, M. G., The Geometry of n Dimensions, C. Griffin and Co., London, 1961.

(19) Kendall, M. G. and Stuart, A., The Advanced Theory of Statistics, Nos. I, II, and III, C. Griffin & Co., London, 1967.

(20) King, B., "Stepwise Clustering Procedures," Journal of the American Statistical Association, Vol. 62, 1967.

(21) Kruskal, J. B., "Multidimensional Scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika, 29, 1964..

(22) Kruskal, J. B., "Nonmetric multidimensional scaling: a numerical method," Psychometrika, 29, 1964.

(23) Ling, R. F., "Cluster Analysis," Technical Report No. 18, Yale University, Department of Statistics, 1971.

(24) Lingoes, J. C. and Guttman, L., "Nonmetric Factor Analysis: A rank reducing alternative to linear factor analysis," Multivariate Behavioral Research, 2, 1967, p. 485-505.

(25) Miller, R. L. and Kahn, J. S., Statistical Analysis in the Geological Sciences, John Wiley and Sons, New York, 1962.

(26) Morrison, D., Multivariate Statistical Methods, McGraw Hill, New York, 1967.

(27) Neyman, J. and Scott, E., "Clustering of Galaxies," Third Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1956.

(28) Pearson, K., Tables of the Incomplete Beta Function, Cambridge University Press, Cambridge, 1968.

(29) Pearson, K., Tables of the Incomplete Gamma Function, Cambridge University Press, Cambridge, 1946.

(30) Pearson, K., "On Lines and Planes of Closest Fit," Phil. Mag., 6, 1901.

(31) Penn, L., An On-Line Statistical Computer System for Lay Usage, Technical Report No. 68, University of Georgia, Department of Statistics, 1971.

(32) Roy, S. N., Some Aspects of Multivariate Analysis, John Wiley & Sons, New York, 1957.

(33) Sokal, R. R. and Michener, C. D., "A Statistical Method for evaluating Systematic relationships," University of Kansas Sci. Bull., 38, 1958.

(34) Sokal, R. R. and Sneath, P. H., Principles of Numerical Taxonomy, W. H. Freeman, San Fransisco and Loneon, 1963.

(35) Stephenson, W., "Inverted Factor Technique," British Journal of Psychology, 1936.

(36) Thurstone, L. L., Multiple Factor Analysis, University of Chicago Press, 1947.

(37) Tryon, R. C., Cluster Analysis, Edwards Bros., Ann Arbor, 1939.

(38) Williams, W. T. and Lambert, J. M., "Multivariate Methods in Plant Ecology," I. J. Ecology, 47, 1959.

APPENDIX A

Source Listings for CLUSTR Program

```
      COMPILER OPTIONS - NAME= MAIN,OPT=00,LINECNT=55,SIZE=0000K,
                         SOURCE,BCD,NOLIST,NODECK,LOAD,MAP,NOLEDIT,NOID,NOXREF
ISN 0002              IMPLICIT REAL*8(D)
ISN 0003              DOUBLE PRECISION BETAX,GY,ARG,DN,DN,OF,TALF,YCRHX,OLOGK,BETAP
ISN 0004              DOUBLE PRECISION GAR,GRS,GRT
ISN 0005              DIMENSION SETLP350(,AX1275(,FMX1000(,F350(,RIAX1275(,MX50(,DS50(,FFCAN C07
                     1G(1000),FRIN(400),RKELI(20),NTEMP(20),KSET(6),CRNT(40)
ISN 0006              DIMENSION DX(20),W(1275),ICOT(50),ISX(50),IGL(50),CLL(50,20),ZZ(20)
                     1, VAR(20),COSR(20),FAI(20),SPROD(40),AAT(50),RNSQLE(50,10)
ISN 0007              COMMON COSUP(20,20),SXSUP(20),OSPROD(40),AX(50,20),NV,NP,NG,
                     1IC(100),NIG(20),NCO(20),NUR(20)
ISN 0008              REWIND 14
ISN 0009              WRITE(6,298)
ISN 0010            298 FORMAT(1*1*)
ISN 0011              READ(5,50) NP,NV,ALFCOR,ALFEXT,AC2,AE2,AC3,AE3
ISN 0012             50 FORMAT(I3,2X,I3,6(1X,F4,0))
ISN 0013              WRITE(6,51) NP,NV
ISN 0014             51 FORMAT(* NP=*,I3,10X,*NV=*,I3)
ISN 0015              READ(5,100) FMT
ISN 0016            100 FORMAT(20A4)
ISN 0017              DO 1 I=1,NP
ISN 0018              READ(5,FMT) (X(I,J),J=1,NV)
ISN 0019              WRITE(14,105) (X(I,J),J=1,NV)
ISN 0020            105 FORMAT(1PSE16.6)
ISN 0021              1 CONTINUE
ISN 0022              REWIND 14
ISN 0023              JJJ=0
ISN 0024            444 JJJ=JJJ+1
ISN 0025              IF (JJJ.EQ.4) GO TO 811
ISN 0027              WRITE(6,111) JJJ
ISN 0028            111 FORMAT(*1PASS NO. *,I2)
ISN 0029              IF(JJJ.EQ.2) ALFCOR=AC2
ISN 0031              IF(JJJ.EQ.2) ALFEXT=AE2
ISN 0033              IF(JJJ.EQ.3) ALFCOR=AC3
ISN 0035              IF(JJJ.EQ.3) ALFEXT=AE3
ISN 0037              WRITE(6,52) ALFCOR,ALFEXT
ISN 0038             52 FORMAT(* ALFCOR=*,F6.3,5X,*ALFEXT=*,F6.3)
ISN 0039              IF(JJJ.EQ.1) GO TO 445
ISN 0041              CALL CCR2
ISN 0042              GO TO 446
ISN 0043            445 CALL CCR1
ISN 0044            446 CONTINUE
ISN 0045              NVV=NV
ISN 0046              NVV=(NV*(NV+1))/2
ISN 0047              WRITE(6,110)
ISN 0048            110 FORMAT(* PSP IS UPPER TRI PART OF METRIC USED FOR STANDARDIZATION*
                     1)
ISN 0049              DO 2 I=1,NVV
ISN 0050              IF(JJJ.EQ.1) GO TO 447
ISN 0052              OSPROD(I)=OSPROD(I)/(RNTG-NG)
ISN 0053              GO TO 448
ISN 0054            447 OSPROD(I)=OSPROD(I)/(NP-1)
ISN 0055            448 CONTINUE
ISN 0056              3 WRITE(6,300) I,OSPROD(I)
ISN 0057            300 FORMAT(* MSPI*,I3,*I= *,F12.4)
ISN 0058              CALL CMPSC(OSPROD,NV,1.D-10,IER)
ISN 0059              DO 4 I=1,NP
```

```
ISN 0060          DO 5 J=1,NV
ISN 0061          CCX=X(I,J)
ISN 0062        5 DV(J)=CCX-DXSUM(J)
ISN 0063          CALL DMTDS(DV,1,NV,DSPROD,-1,JER)
ISN 0064          DSUM=0.D0
ISN 0065          DO 6 J=1,NV
ISN 0066        6 DSUM=DSUM+DV(J)**2
ISN 0067          DO 7 J=1,NV
ISN 0068          DPX=DV(J)/DSQRT(DSUM)
ISN 0069          X(I,J)=DPX
ISN 0070        7 CONTINUE
ISN 0071        4 CONTINUE
ISN 0072          DO 10 J=1,NP
ISN 0073          DO 11 I=1,J
ISN 0074          KI=KDE(I,J)
ISN 0075          W(KI)=0.0
ISN 0076          DO 12 L=1,NV
ISN 0077       12 W(KI)=W(KI)+X(I,L)*X(J,L)
ISN 0078       11 CONTINUE
ISN 0079       10 CONTINUE
ISN 0080          DO 20 I=1,NF
ISN 0081       20 ICL(I)=0
ISN 0082          DO 21 I=1,NP
ISN 0083          DO 22 J=1,2C
ISN 0084       22 CL(I,J)=0.0
ISN 0085       21 CONTINUE
ISN 0086          KCL=1
ISN 0087       97 I1=0
ISN 0088          DO 23 I=1,NP
ISN 0089          IEXT(I)=0
ISN 0090       23 ICON(I)=0
ISN 0091          WMAX=0.0
ISN 0092          NPM=NP-1
ISN 0093          DO 24 I=1,NPM
ISN 0094          IPL=I+1
ISN 0095          DO 25 J=IPL,NP
ISN 0096          IF((ICL(I).EQ.1).OR.(ICL(J).EQ.1)) GO TO 25
ISN 0097          IF(W(KDE(I,J)).LT.WMAX) GO TO 25
ISN 0100          WMAX=W(KDE(I,J))
ISN 0101          I1 = I
ISN 0102          I2 = J
ISN 0103       25 CONTINUE
ISN 0104       24 CONTINUE
ISN 0105          WRITE(6,53) I1,I2,WMAX
ISN 0106       53 FORMAT('OPTS ',I3,' AND ',I3,' ARE CLOSEST WITH CORR ',F12.8)
ISN 0107          RMAX=WMAX
ISN 0108          ALF=ICOS(RMAX,2)
ISN 0109          WRITE(6,55) ALF
ISN 0110       55 FORMAT(' ALF= ',F12.8)
ISN 0111          IF(ALF.GT.ALFCOR) GO TO 95
ISN 0113          ICL(I1)=1
ISN 0114          ICL(I2)=1
ISN 0115          ICON(I)=I1
ISN 0116          CL(I1,KCL)=ALF
ISN 0117          CL(I2,KCL)=ALF
ISN 0118          LPT=I2
```

```
ISN 0119          LPX=LPT
ISN 0120           ENSC=1.0
ISN 0121          KK=1
ISN 0122       35 KK=KK+1
ISN 0123          WSUM=0.0
ISN 0124          KKM=KK-1
ISN 0125          DO 26 J=1,KKM
ISN 0126       26 WSUM=WSUM+W(KC(ICOR(J),LPT))
ISN 0127           ENSC= ENSC+1.0+2.0*WSUM
ISN 0128          ICOR(KK)=LPT
ISN 0129       36 SUMMAX=-30.0
ISN 0130          DO 29 I=1,NP
ISN 0131          DO 28 J=1,KK
ISN 0132          IF(I.EQ.ICOR(J)) GO TO 29
ISN 0134       28 CONTINUE
ISN 0135          DO 38 J=1,NP
ISN 0136          IF(I.EQ.IEXT(J)) GO TO 29
ISN 0138       38 CONTINUE
ISN 0139          CNSUM=0.0
ISN 0140          DO 31 KX=1,KK
ISN 0141       31 CNSUM=CNSUM+W(KC(ICOR(KX),I))
ISN 0142          IF(CNSUM.LT.SUMMAX) GO TO 32
ISN 0144          SUMMAX=CNSUM
ISN 0145          LPT=I
ISN 0146       32 CONTINUE
ISN 0147       29 CONTINUE
ISN 0148          IF(SUMMAX.LE.0.0) GO TO 56
ISN 0150          RMAX=SUMMAX/SQRT( ENSC)
ISN 0151          KKK=KK+1
ISN 0152          WRITE(6,56) KKK,LPT,RMAX
ISN 0153       56 FORMAT('0POINT ',I3,' IN CLUSTER IS POINT ',I3,'.  CORR IS',F8.4)
ISN 0154          ALF=TEST(RMAX,KKK)
ISN 0155          WRITE(6,57) ALF
ISN 0156       57 FORMAT(' ALF= ',F8.4)
ISN 0157          IF(ALF.LE.ALFCCR) GO TO 33
ISN 0159          IF(KK.EQ.2) ICL(LPX)=2
ISN 0161          IF((ALF.GT.ALFCCR).AND.(ALF.LE.ALFEXT)) GO TO 34
ISN 0163          GO TO 95
ISN 0164       33 ICL(LPT)=1
ISN 0165          CL(LPT,KCL)=ALF
ISN 0166          GO TO 35
ISN 0167       34 IF(ICL(LPT).EQ.1) GO TO 37
ISN 0169          ICL(LPT)=2
ISN 0170       37 CONTINUE
ISN 0171          CL(LPT,KCL)=ALF
ISN 0172          IE=IE+1
ISN 0173          IEXT(IE)=LPT
ISN 0174          GO TO 36
ISN 0175       96 KCL=KCL+1
ISN 0176          IF(KCL.EQ.21) GO TO 95
ISN 0178          GO TO 97
ISN 0179       95 CONTINUE
ISN 0180          DO 82 KX=1,2
ISN 0181          JA=10*(KX-1)+1
ISN 0182          JC=JA+9
ISN 0183          WRITE(6,45)
```

```
ISN 0184        45 FORMAT('1','PT NO.',32X,'CLUSTERS')
ISN 0185           IF(KK.EQ.2) GO TO 113
ISN 0187           WRITE(6,112)
ISN 0188        112 FORMAT(13X,'NC.  1',4X,'NC.  2',4X,'NO.  3',4X,'NO.  4',4X,
                  1           'NC.  5',4X,'NC.  6',4X,'NO.  7',4X,'NO.  8',4X,
                  2           'NC.  9',4X,'NO. 10')
ISN 0189           GO TO 115
ISN 0190        113 WRITE(6,114)
ISN 0191        114 FORMAT(13X,'NC. 11',4X,'NC. 12',4X,'NC. 13',4X,'NO. 14',4X,
                  1           'NC. 15',4X,'NC. 16',4X,'NO. 17',4X,'NO. 18',4X,
                  2           'NC. 19',4X,'NC. 20')
ISN 0192        115 CONTINUE
ISN 0193           WRITE(6,46)
ISN 0194         46 FORMAT('0')
ISN 0195           DO 47 I=1,NP
ISN 0196         47 WRITE(6,48) I,       (CL(I,J),J=JA,JB)
ISN 0197         48 FORMAT(' ',1X,I3,6X   ,10(2X,F8.6))
ISN 0198         88 CONTINUE
ISN 0199           DO 401 I=1,100
ISN 0200        401 IG(I)=0
ISN 0201           DO 402 I=1,20
ISN 0202        402 NCO(I)=0
ISN 0203           DO 403 J=1,20
ISN 0204           DO 404 I=1,NP
ISN 0205           IF((CL(I,J).GT.0.0).AND.(CL(I,J).LE.ALFCOR)) GO TO 405
ISN 0207           GO TO 404
ISN 0208        405 NCO(J)=NCO(J)+1
ISN 0209        404 CONTINUE
ISN 0210        403 CONTINUE
ISN 0211           DO 407 I=1,20
ISN 0212        407 NCR(I)=0
ISN 0213           DO 409 K=1,20
ISN 0214           NMAX=-1
ISN 0215           DO 406 I=1,20
ISN 0216           DO 408 J=1,20
ISN 0217           IF(I.EQ.NGR(J)) GO TO 408
ISN 0219        408 CONTINUE
ISN 0220           IF(NCO(I).LE.NMAX) GO TO 406
ISN 0222           NMAX=NCO(I)
ISN 0223           IZ=I
ISN 0224        406 CONTINUE
ISN 0225           NGR(K)=IZ
ISN 0226        409 CONTINUE
ISN 0227        411 CONTINUE
ISN 0228           J=0
ISN 0229           KPT=0
ISN 0230        419 J=J+1
ISN 0231           KPT=KPT+1
ISN 0232           IF(NCO(NGR(J)).LE.2) GO TO 99
ISN 0234           ICT=0
ISN 0235           DO 413 NZ=1,NP
ISN 0236           IF(CL(NZ,NGR(J)).GT.0.0) GO TO 414
ISN 0238           GO TO 413
ISN 0239        414 ICT=ICT+1
ISN 0240           IF((CL(NZ,NGR(J)).LE.ALFCOR).AND.(IG(NZ).EQ.0)) IG(NZ)=KPT
ISN 0242        413 CONTINUE
```

```
ISN 0243            IF((JJJ.EQ.2).OR.(JJJ.EQ.3)) GO TO 502
ISN 0245            DO 415 JX=1,20
ISN 0246            IA=0
ISN 0247            IF(NOR(J).EQ.NOR(JX)) GO TO 415
ISN 0249            DO 416 I=1,AP
ISN 0250            IF((CL(I,NOR(J)).NE.0.0).AND.(CL(I,NOR(JX)).NE.0.0))IMATC=IMATC+1
ISN 0252          416 CONTINUE
ISN 0253            ICTT=ICT-2
ISN 0254            IF((IMATC.GT.2).AND.(IMATC.LE.ICTT)) GO TO 417
ISN 0256            GO TO 415
ISN 0257          417 DO 418 IXX=1,AP
ISN 0259            IF((CL(IXX,NOR(JX)).LE.ALFCOR).AND.(CL(IXX,NOR(JX)).GT.0.0)).AND.
                   1(IG(IXX).EQ.0)) IG(IXX)=KPI
ISN 0260          418 CONTINUE
ISN 0261          415 CONTINUE
ISN 0262          502 CONTINUE
ISN 0263            NC=0
ISN 0264            DO 420 IQ=1,AP
ISN 0265            IF(IG(IQ).EQ.KPI) NC=NC+1
ISN 0267          420 CONTINUE
ISN 0268            IF(NC.GT.2) GO TO 999
ISN 0270            DO 421 IQ=1,AP
ISN 0271            IF(IG(IQ).EQ.KPI) IG(IQ)=0
ISN 0273          421 CONTINUE
ISN 0274          999 CONTINUE
ISN 0275            GO TO 439
ISN 0276           99 CONTINUE
ISN 0277            DO 425 I=1,20
ISN 0278          425 NIG(I)=0
ISN 0279            DO 422 I=1,20
ISN 0280            DO 423 J=1,AP
ISN 0281            IF(IG(J).EQ.I) NIG(I)=NIG(I)+1
ISN 0283          423 CONTINUE
ISN 0284          422 CONTINUE
ISN 0285            WRITE(6,430) (NIG(I),I=1,20)
ISN 0286          430 FORMAT(' ',20(I2,2X))
ISN 0287            NG=0
ISN 0288            DO 424 I=1,AP
ISN 0289            IF(IG(I).LT.NG) GO TO 424
ISN 0291            NG=IG(I)
ISN 0292          424 CONTINUE
ISN 0293            NNIG=0
ISN 0294            DO 501 I=1,20
ISN 0295          501 NNIG=NNIG+NIG(I)
ISN 0296            WRITE(6,552) NG,NNIG
ISN 0297          552 FORMAT('ONG. OF GROUPS FORMED=',I2,5X,'NO. OF PTS ASSIGNED TO GROU
                   1PS= ',I3)
ISN 0298            DO 551 I=1,AP
ISN 0299          551 READ(14,105) (X(I,J),J=1,AV)
ISN 0300            REWIND 14
ISN 0301            GO TO 444
ISN 0302          811 CONTINUE
ISN 0303            REWIND 15
ISN 0304            CALL CCR2
ISN 0305            NVV = (NV*(NV+1))/2
ISN 0306            DO 10000 I=1,NVV
```

```
ISN 0307          DSPROD(I) = DSPROD(I)/(ANIG-NG)
ISN 0308          WRITE (6,300) I,DSPROD(I)
ISN 0309    10000 CONTINUE
ISN 0310          DO 11010 I=1,NVV
ISN 0311    11010 VARX(I) = DSPRO (I)
ISN 0312          DO 20000 I=1,NG
ISN 0313          IF (NIG(I).EQ.0) GO TO 20000
ISN 0315          DO 20000 J=1,NV
ISN 0316          DGSUM(I,J) = DGSUM(I,J)/NIG(I)
ISN 0317    20000 CONTINUE
ISN 0318          DO 20001 I=1,NG
ISN 0319          WRITE(6,11020) (DGSUM(I,J),J=1,NV)
ISN 0320          WRITE(7,11020) (DGSUM(I,J),J=1,NV)
ISN 0321    20001 CONTINUE
ISN 0322    11020 FORMAT(14X,6E7.3)
ISN 0323          CALL DNFSQ(DSPROD,NV,1.D-10,IER)
ISN 0324          DO 10001 I=1,NP
ISN 0325          IF (IG(I).EQ.0) GO TO 10003
ISN 0327          KK = IG(I)
ISN 0328          DO 10002 J=1,NV
ISN 0329          DDX = X(I,J)
ISN 0330    10002 DV(J) = DDX - DGSUM(KK,J)
ISN 0331          GO TO 10005
ISN 0332    10003 DO 10004 J=1,NV
ISN 0333          DDX = X(I,J)
ISN 0334    10004 DV(J) = DDX - DXSUM(J)
ISN 0335    10005 CALL DNTOS(DV,1,NV,DSPROD,-1,IER)
ISN 0336          DO 10007 J=1,NV
ISN 0337          X(I,J) = DV(J)
ISN 0338    10007 CONTINUE
ISN 0339          WRITE(15,2) I,(X(I,J),J=1,NV)
ISN 0340    10001 CONTINUE
ISN 0341          NVV = ENV*(NV+1)/2
ISN 0342          DO 11000  I=1,NVV
ISN 0343    11000 SPROD(I) = DSPROD(I)
ISN 0344          NVV = NV**2
ISN 0345          REWIND 15
ISN 0346          KFACT = NV
              C   USE WITH SUBROUTINES FACT,WRIR,LTERM,INSLD                    FCAN 006
ISN 0347          NIN=5                                                         FCANA006
ISN 0348          NOUT=6                                                        FCANB006
ISN 0349          CALF = 0.01D0
              C   LIMITATION... 50 VARIABLES,  20 FACTORS                      FCAN 009
              C   READ HEADER CARD                                            FCAN 011
ISN 0350          DO 17 KARD = 1,NP
ISN 0351          READ (15,2) IC,(SETUP(J),J=1,KFACT)
ISN 0352        2 FORMAT(1X,I2,3X,7F10.7)
ISN 0353          DO 13 J=1,KFACT
ISN 0354          IX = IJ-1+NP + IC
ISN 0355       13 PH(IX) = SETUP(J)
ISN 0356       17 CONTINUE
ISN 0357          DO 14 K=1,NF
ISN 0358          H2K = 0.
ISN 0359          DO 14 J=1,KFACT
ISN 0360          KX = IJ-1+NP + K
ISN 0361       14 H(K) = H(K) + PH(KX)**2
```

```
ISN 0362          DO 1002  J=1,NP
ISN 0363          IF(PHI(J)) 1003,1003,1004                                  FCAN 195
ISN 0364     1003 D(J) = 0.                                                  FCAN 196
ISN 0365          GO TO 1002                                                 FCAN 197
ISN 0366     1004 D(J) = SQRT (PHI(J))                                       FCAN 198
ISN 0367     1002 CONTINUE                                                   FCAN 199
ISN 0368          ASOL = 0                                                   FCAN 200
ISN 0369          DM = DFLOAT(KFACT-1)/2.00
ISN 0370          SNTHTA = DSQRT((BETAP(6)-CO)/DFLOAT(NP-KFACT+1),0.500,DM))
ISN 0371          IF (SNTHTA.LT.0.1) SNTHTA = 0.1
ISN 0372          DO 1001 I=1,NP                                             FCAN 201
ISN 0373          IF(PHI(I)) 1001,1001,1005                                  FCAN 202
ISN 0374     1005 ICO = 0                                                    FCAN 203
ISN 0375          DO 1006 K = 1,KFACT                                        FCAN 204
ISN 0376          IALP = IK-1)*NP + I                                        FCAN 205
ISN 0377     1006 PRIM(K) = PHI(IALP)/D(I)                                   FCAN 206
ISN 0378          BNO = 0.                                                   FCAN 207
ISN 0379     1021 DO 1005 J=1,NP                                             FCAN 208
ISN 0380          SUM = 0.                                                   FCAN 209
ISN 0381          DO 1007 K=1,KFACT                                          FCAN 210
ISN 0382          IALP = IK-1)*NP + J                                        FCAN 211
ISN 0383     1007 SUM = SUM + PHI(IALP)*PRIM(K)                              FCAN 212
ISN 0384          FC(J) = SUM                                                FCAN 213
ISN 0385          D(J) = AP + J                                              FCAN 214
ISN 0386          IF(ABS(FC(J)) .GE. 1.E8 * D(J) ) GO TO 1011
ISN 0387          CAT = ABS(AINT(SNGL(10.*FC(J)/D(J)))*0.1/SNTHTA))
ISN 0388          IF(CAT - BAD) 1010,1011,1011                               FCAN 216
ISN 0389     1011 FC(J) = 0.                                                 FCAN 217
ISN 0390          GO TO 1005                                                 FCAN 218
ISN 0391     1010 FC(J) = SQRT (BAD - CAT)                                   FCAN 219
ISN 0392     1005 CONTINUE                                                   FCAN 220
ISN 0393          IF(IGO) 1060,1060,2008                                     FCAN 221
ISN 0394     1060 NCZE = 0                                                   FCAN 222
ISN 0395          DO 1081 J=1,NP                                             FCAN 223
ISN 0396          JAT = NP+J                                                 FCAN 224
ISN 0397          IF(FC(JAT)) 1081,1082,1081                                 FCAN 225
ISN 0398     1082 NCZE = NCZE + 1                                            FCAN 226
ISN 0399     1081 CONTINUE                                                   FCAN 227
ISN 0400          IF(NCZE - 2) 1083,1083,2008                                FCAN 228
ISN 0401     1083 BNO = BNO - 1.                                             FCAN 229
ISN 0402          GO TO 1021                                                 FCAN 230
ISN 0403     2008 IF(IGO - 7) 2013,1020,1020                                 FCAN 231
ISN 0404     2013 TNOR = 0.                                                  FCAN 232
ISN 0405          DO 1013 K=1,KFACT                                          FCAN 233
ISN 0406          AV = 0.                                                    FCAN 234
ISN 0407          ASQ = 0.                                                   FCAN 235
ISN 0408          DO 1014 J=1,NP                                             FCAN 236
ISN 0409          JALP = IK-1)*NP + J                                        FCAN 237
ISN 0410          JAT = NP + J                                               FCAN 238
ISN 0411          AV = AV + PHI(JALP)*FC(J)*FC(JAT)                          FCAN 239
ISN 0412     1014 ASQ = ASQ + PHI(JALP)*PHI(JALP)*FC(JAT)                    FCAN 240
ISN 0413          IF (ASQ .EQ. 0.) GO TO 1013
ISN 0414          E = AV/ASQ                                                 FCAN 241
ISN 0415          PRIM(K) = (PRIM(K) - E)/21. - E*PRIM(K)                    FCAN 242
ISN 0416     1013 TNOR = TNOR + PRIM(K)*PRIM(K)                              FCAN 243
ISN 0417          TNOR = SQRT (TNOR)                                         FCAN 244
```

```
ISN 0421          DO 1016 K=1,KFACT                                          FCAN 245
ISN 0422       1016 FRIN=XK < * FRIN=XK/TNOR                                 FCAN 246
ISN 0423          ICO = IGO + 1                                              FCAN 247
ISN 0424          GO TO %1017,1018,1019,1019,1019,1019,1019<,IGO             FCAN 248
ISN 0425       1017 BND = 3.                                                 FCAN 249
ISN 0426          GO TO 1021                                                 FCAN 250
ISN 0427       1018 BND = 2.                                                 FCAN 251
ISN 0428          GO TO 1021                                                 FCAN 252
ISN 0429       1019 BND = 1.                                                 FCAN 253
ISN 0430          GO TO 1021                                                 FCAN 254
ISN 0431       1020 NCON = 0                                                 FCAN 255
ISN 0432          VMAX = C.0
ISN 0433          VMIN = 0.0
ISN 0434          DO 1121 J=1,NP                                             FCAN 256
ISN 0435          IF(ZERO(J)< 1121,1121,1022                                 FCAN 257
ISN 0436       1022 RAT(J) = FO(J)/C(J)
ISN 0437          IF (ABS(RAT(J))-SNTHTA) 1023,1023,1121
ISN 0438       1023 NCON = NCON + 1
ISN 0439          IF (VMAX.LE.RAT(J)) VMAX = RAT(J)
ISN 0441          IF (VMIN.GE.RAT(J)) VMIN = RAT(J)
ISN 0443          RAT(J) = C.C
ISN 0444       1121 CONTINUE
ISN 0445          IF (NCON.LE.KFACT+2) GO TO 1001
ISN 0447          DELTA = (ARSIN(VMAX)-ARSIN(VMIN))/2.0
ISN 0448          ARG = (SIN(DELTA))**2
ISN 0449          DN = DFLOAT(KFACT-1)/2.CO
ISN 0450          DY = BETAX(ARG,0.500,DN)
ISN 0451          DN = DFLOAT(NCON-KFACT)<
ISN 0452          DN = DFLOAT(N-NCON)<
ISN 0453          DP = BETAX(DY,DN,DN)<
ISN 0454          IF(DP.GT.CALF< GO TO 1001
ISN 0456       1322 IF(NSOL< 1024,1024,1025                                  FCAN 262
ISN 0457       1025 DO 1026 L=1,NSOL                                         FCAN 263
ISN 0458          SUM = 0.                                                   FCAN 264
ISN 0459          DO 1027 K=1,KFACT                                          FCAN 265
ISN 0460          IAL = NL-1)*KFACT + K                                      FCAN 266
ISN 0461       1027 SUM = SUM + FRIN=XK<*RINV=AL<                            FCAN 267
ISN 0462          IF (ABS(SUM)-C.7) 1026,1001,1001
ISN 0463       1026 CONTINUE                                                 FCAN 269
ISN 0464       1024 DO 1030 K=1,KFACT
ISN 0465          IEL = NSOL*KFACT + K                                       FCAN 271
ISN 0466       1030 RINV=EL< = FRIN=XK<                                      FCAN 272
ISN 0467          NSOL = NSOL + 1                                            FCAN 273
ISN 0468          WRITE (NOUT,1031< NSOL,NCON,DP
ISN 0469       1031 FORMAT(1HO//,15H  PLANE NO. V F3.2,2H  OF SIMPLE STRUCTURE WITH I4,FCAN 275
                   13H ZERO LOADINGS %-.1 LESS V/M LESS (.1<,D12.5/<
ISN 0470          CALL BRIEFC,NP,0<                                          FCAN 277
ISN 0471          WRITE(NOUT,1033< NSOL                                      FCAN 278
ISN 0472       1033 FORMAT(1HO// 75H  VECTOR WHICH TRANSFORMS ORIGINAL FACTOR MATRIX IFCAN 279
                   1NTO THE ABOVE PLANE NO. V I3<                             FCAN 280
ISN 0473          CALL WRIGHT(N,KFACT,0<                                     FCAN 281
ISN 0474          CALL MTDS (FRIN,1,NV,SPROD,-2,JER)
ISN 0475          SS = 0.
ISN 0476          DO 740 JJ=1,NV
ISN 0477        740 SS = SS + FRIN(JJ)**2
ISN 0478          DO 741 JK = 1,NV
```

```
ISN 0479       741 SETUP(JK) = FRIP(JK)/SQRT(SS)
ISN 0480           WRITE (6,790)
ISN 0481       790 FORMAT(1HO,10X,'TRANSFORMATION VECTOR TO TRANSFORM RAW DATA TO
                  1SIMPLE STRUCTURE')
ISN 0482           CALL WRIP(SETUP,KFACT,0)
ISN 0483           WRITL (7,11020) (SETUP(JL),JL=1,KFACT)
ISN 0484           CALL PPPO (FRIM,VARX,Z,1,NV,0,1,NV)
ISN 0485           DO 11003 LL=1,NV
ISN 0486           K = (LL*(LL+1))/2
ISN 0487     11003 CORR(LL) = Z(LL)/SQRT(VARX(K))
ISN 0488           WRITE (6,11004) (CORR(LL),LL=1,NV)
ISN 0489     11004 FORMAT(1HO,7X,'CORRELATIONS BETWEEN ORIGINAL VARIABLES AND THIS
                  1VECTOR ARE'//(1CF12.4))
ISN 0490           DO 11005 LL=1,NP
ISN 0491           (NNSEL(LL,NSEL) = 0
ISN 0492           IF (ABS(LL).GE.0.0) NNSEL(LL,NSEL) = 1
ISN 0494     11005 CONTINUE
ISN 0495      1001 CONTINUE                                                    FCAN 282
ISN 0496           IF(NSEL) 1640,1650,1641                                    FCAN 283
ISN 0497      1642 WRITE(NOUT,1642)                                           FCAN 284
ISN 0498      1642 FORMAT(1H1 21X 25HNO SIMPLE STRUCTURE SOLUTIONS)           FCAN 285
ISN 0499           GO TO 1643                                                 FCAN 286
ISN 0500      1641 DO 1070 J=1,NSOL                                           FCAN 287
ISN 0501           DO 1070 I=1,J                                              FCAN 288
ISN 0502           SUM = 0.                                                   FCAN 289
ISN 0503           DO 1071 K=1,KFACT                                          FCAN 290
ISN 0504           KI = (I-1)*KFACT + K                                       FCAN 291
ISN 0505           KJ = (J-1)*KFACT + K                                       FCAN 292
ISN 0506      1071 SUM = SUM + RINV(KI)*RINV(KJ)                              FCAN 293
ISN 0507           IJ = NOI(I,J)
ISN 0508      1070 RIJ(IJ) = SUM                                              FCAN 295
ISN 0509           WRITE(NOUT,1073)                                           FCAN 296
ISN 0510      1073 FORMAT(1H1 18X 33HCOSINES BETWEEN REFERENCE VECTORS)       FCAN 297
ISN 0511           CALL PRINT(RIJ,NSOL,1)                                     FCAN 298
ISN 0512           WRITE(NOUT,1600)                                           FCAN 299
ISN 0513      1600 FORMAT(1H1 27X 18HSUMMARY OF RESULTS/)                     FCAN 300
ISN 0514           KAE = (NSOL-1)/6 + 1                                       FCAN 301
ISN 0515           IF (  MOD (NSOL,6)                                         FCAN 302
ISN 0516           DO 1602 KA=1,KAE                                           FCAN 303
ISN 0517           IF(KA-KAE) 1604,1606,1606                                  FCAN 304
ISN 0518      1604 KB=6                                                       FCAN 305
ISN 0519           GO TO 1609                                                 FCAN 306
ISN 0520      1606 IF(IC) 1604,1604,1610                                      FCAN 307
ISN 0521      1610 KB=IE                                                      FCAN 308
ISN 0522      1609 DO 1612 KB=1,KBE                                           FCAN 309
ISN 0523      1612 KSET(KB) = KB + 6*(KA-1)                                   FCAN 310
ISN 0524           WRITE(NOUT,1614) (KSET(J),J=1,KBE)                         FCAN 311
ISN 0525      1614 FORMAT(1H0/1GH VAR      V I2, 6H       V I2, 6H       V I2, 6H     FCAN 312
                  1  V I2, 6H       V I2, 6H       V I2, 1H /)                FCAN 313
ISN 0526           DO 1616 KC=1,NP                                            FCAN 314
ISN 0527           DO 1610 KD=1,KBE
ISN 0528           SUM = 0.                                                   FCAN 316
ISN 0529           DO 1620 KG=1,KFACT                                         FCAN 317
ISN 0530           MKI = (KG-1)*NP + KC                                       FCAN 318
ISN 0531           MJJ = (KA-1)*6 + KDI - 1)*KFACT + KG
ISN 0532      1620 SUM = SUM + FM(MKI)*RINV(MJJ)                              FCAN 320
```

```
ISN 0535      1618 SETUP(KC) = SUM
ISN 0534           WRITE(6,1624) KC,(SETUP(J),J=1,KC)                    FCAN 322
ISN 0535      1624 FORMAT(1H0,6F10.4)                                     FCAN 323
ISN 0536      1616 CONTINUE                                               FCAN 324
ISN 0537      1602 CONTINUE                                               FCAN 325
ISN 0538           DO 1644 I=1,NP
ISN 0539           READ (14,105)(X(I,J),J=1,NV)
ISN 0540      1644 CONTINUE
ISN 0541           REWIND 14
ISN 0542           IF (NSOL.GE.10) GO TO 1646
ISN 0543           L = NSOL + 1
ISN 0544           DO 1645 I=L,10
ISN 0545           DO 1645 KK=1,NP
ISN 0546           INNSOL(KK,I) = 0
ISN 0547      1645 CONTINUE
ISN 0548      1646 DO 1647 I=1,NP
ISN 0549           WRITE (7,1650) I,IG(I),(INNSOL(I,J),J=1,10),(X(I,J),J=1,NV)
ISN 0550      1647 CONTINUE
ISN 0551      1650 FORMAT(1X,I2,I1,10I1,8F7.3)
ISN 0552           WRITE (6,1760)
ISN 0553      1760 FORMAT(1H1)
ISN 0554           WRITE (6,1790)
ISN 0555      1790 FORMAT(41X,'PLANES')
ISN 0556           WRITE (6,1791)
ISN 0557      1791 FORMAT(10X,'POINT #',1X,'CLUSTER #',5X,'1',1X,'2',1X,'3',1X,'4',1X
                  1,'5',1X,'6',1X,'7',1X,'8',1X,'9',1X,'10',10X,'COORDINATES(OBSERVAT
                  2IONS)')
ISN 0558           DO 1710 I=1,NP
ISN 0559           WRITE (6,1752) I,IG(I),(INNSOL(I,J),J=1,10),(X(I,J),J=1,NV)
ISN 0560      1710 CONTINUE
ISN 0561      1752 FORMAT(14X,I2,6X,I1,10X,10(I1,1X),5X,8(F7.3,2X))
ISN 0562      1648 STOP
ISN 0563           END                                                    FCAN 349
```

```
LEVEL 19 ( JUNE 70 )                    OS/360  FORTRAN H                    DATE  71.257/21.42.06

          COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=58,SIZE=0000K,
                             SOURCE,BCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF
  ISN 0002            FUNCTION KD(I,J)
  ISN 0003            IF(I.LT.J) GO TO 1
  ISN 0005            KD=(I*(I-1))/2+J
  ISN 0006            GO TO 99
  ISN 0007          1 KD=(J*(J-1))/2+I
  ISN 0008         99 RETURN
  ISN 0009            END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=58,SIZE=0000K,
                           SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF

                CHRIR                                                          042
                C       SUBROUTINE FOR WRITING CORRELATIONS, MEANS, AND FACTOR LOADINGS  043
      ISN 0002          SUBROUTINE WRITER(R,NV,NN)                             044
                C       NN IS ZERO FOR FACTOR LOADINGS AND MEANS, ONE OTHERWISE 045
      ISN 0003          NIN=5                                                  A045
      ISN 0004          NOUT=6                                                 B045
      ISN 0005          DIMENSION R(1275),JWR(10),RWR(10)                      046
      ISN 0006          WRITE(NOUT,1183)                                       047
      ISN 0007          DO 1101 I=1,NV                                         048
      ISN 0008          NRE = (NV-1)/10 + 1
      ISN 0009          DO 1102 J=1,NRE                                        050
      ISN 0010          IF(10*J-NV) 1103,1103,1104
      ISN 0011       1103 K=1
      ISN 0012          GO TO 1105                                             053
      ISN 0013       1104 K=10-10*J+NV
      ISN 0014       1105 DO 1107 L=1,K                                        055
      ISN 0015       1107 JWR(L) = 10*J-1+L
      ISN 0016          WRITE(NOUT,1181) (JWR(M),M=1,K)                        057
      ISN 0017       1181 FORMAT(1HO 13X I2,9(9X,I2))
      ISN 0018          IF(NN) 1129,1130,1129                                  059
      ISN 0019       1130 DO 1131 M=1,K                                        060
      ISN 0020          MR = JWR(M)                                            061
      ISN 0021       1131 RWR(M) = R(MR)                                       062
      ISN 0022          WRITE(NOUT,1185) (RWR(N),N=1,K)                        063
      ISN 0023       1185 FORMAT(7X 10F11.5)
      ISN 0024          GO TO 1102                                             065
      ISN 0025       1129 DO 1120 M=1,K                                        066
      ISN 0026          MR = NC(I,JWR(M))
      ISN 0027       1120 RWR(M) = R(MR)                                       068
      ISN 0028          WRITE(NOUT,1182) I,(RWR(N),N=1,K)                      069
      ISN 0029       1182 FORMAT(5H ROW ,I2,10F11.5)
      ISN 0030       1102 CONTINUE                                             071
      ISN 0031          IF(NN) 1127,1128,1127                                  072
      ISN 0032       1127 WRITE(NOUT,1183)                                     073
      ISN 0033       1183 FORMAT(1HO)                                          074
      ISN 0034       1101 CONTINUE                                             075
      ISN 0035       1128 RETURN                                               076
      ISN 0036            END                                                  077
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=58 SIZE=0000K,
                           SOURCE,BCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF
ISN 0002            SUBROUTINE CORR
ISN 0003            IMPLICIT REAL*8(D)
ISN 0004            COMMON DCSUM(20,20),DXSUM(20),DSPROD(400),X(50,20),NV,NP,NG,
                   1IG(100),NIG(20),NCO(20),NCR(20)
ISN 0005            NC=(NV*(NV+1))/2
ISN 0006            DO 1 I=1,NV
ISN 0007          1 DXSUM(I)=0.00
ISN 0008            DO 2 I=1,NC
ISN 0009          2 DSPROD(I)=0.00
ISN 0010            DO 3 I=1,NV
ISN 0011            DO 4 J=1,NP
ISN 0012            DX=X(J,I)
ISN 0013          4 DXSUM(I)=DXSUM(I)+DX
ISN 0014            DXSUM(I)=DXSUM(I)/(DFLOAT(NP))
ISN 0015            WRITE(6,200) I,DXSUM(I)
ISN 0016        200 FORMAT(' MEAN OF VAR(',I2,')= ',F9.4)
ISN 0017          3 CONTINUE
ISN 0018            WRITE(6,300)
ISN 0019        300 FORMAT(' DSPROD IS UPPER TRI PART OF MATRIX OF CORRECTED S.S. AND
                   1S.P. FOR NV VARS. STORED BY COLS')
ISN 0020            DO 5 J=1,NV
ISN 0021            DO 6 I=1,J
ISN 0022            KI=KO(I,J)
ISN 0023            DO 7 L=1,NP
ISN 0024            DX1=X(L,J)
ISN 0025            DX2=X(L,I)
ISN 0026          7 DSPROD(KI)=DSPROD(KI)+DX1*DX2
ISN 0027            DSPROD(KI)=DSPROD(KI)-DXSUM(I)*DXSUM(J)*DFLOAT(NP)
ISN 0028            WRITE(6,100) KI,DSPROD(KI)
ISN 0029        100 FORMAT(' DSPROD(',I3,')= ',F13.4)
ISN 0030          6 CONTINUE
ISN 0031          5 CONTINUE
ISN 0032            RETURN
ISN 0033            END
```

```
          COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=5..,SIZE=0000K,
                             SOURCE,EBCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF
  ISN 0002            SUBROUTINE CORZ
  ISN 0003            IMPLICIT REAL*8(D)
  ISN 0004            COMMON DGSUM(20,20),DXSUM(20),DSPROD(400),X(50,20),NV,NP,NG,
                     1IG(100),NIG(20),KCO(20),NOB(20)
  ISN 0005            NC=(NV*(NV+1))/2
  ISN 0006            DO 2 I=1,NC
  ISN 0007          2 DSPROD(I)=0.DO
  ISN 0008            DO 11 I=1,NG
  ISN 0009            DO 12 J=1,NV
  ISN 0010         12 DGSUM(I,J)=0.D0
  ISN 0011         11 CONTINUE
  ISN 0012            DO 13 K=1,NG
  ISN 0013            DO 3 I=1,NV
  ISN 0014            DO 4 J=1,NP
  ISN 0015            DX=X(J,I)
  ISN 0016            IF(IG(J).NE.K) GO TO 4
  ISN 0017            DGSUM(K,I)=DGSUM(K,I)+DX
  ISN 0018          4 CONTINUE
  ISN 0019          3 CONTINUE
  ISN 0020         13 CONTINUE
  ISN 0021            DO 57 I=1,NV
  ISN 0022            DO 58 K=1,NG
  ISN 0023            WRITE(6,159) I,K,DGSUM(K,I)
  ISN 0024        159 FORMAT(' SUM OF OBSERS. FOR VAR',I2,' IN GROUP',I2,'= ',D12.4)
  ISN 0025         58 CONTINUE
  ISN 0026         57 CONTINUE
  ISN 0027            WRITE(6,300)
  ISN 0028        300 FORMAT(' DSPROD IS UPPER TRI PART OF MATRIX OF CORRECTED S.S. AND
                     1S.P. FOR NV VARS, STORED BY-COLS')
  ISN 0029            DO 5 J=1,NV
  ISN 0030            DO 6 I=1,J
  ISN 0031            KI=KO(I,J)
  ISN 0032            DO 7 L=1,NP
  ISN 0033            IF(IG(L).EQ.0) GO TO 7
  ISN 0034            DXI=X(L,I)
  ISN 0035            DX2=X(L,J)
  ISN 0036            DSPROD(KI)=DSPROD(KI)+DXI*DX2
  ISN 0037          7 CONTINUE
  ISN 0038            DCOR=0.DO
  ISN 0039            DO 14 L=1,NG
  ISN 0040            IF(NIG(L).EQ.0) GO TO 23
  ISN 0041            DCOR=DCOR+DGSUM(L,I)*DGSUM(L,J)/NIG(L)
  ISN 0042            GO TO 14
  ISN 0043         23 NC=NC-1
  ISN 0044         14 CONTINUE
  ISN 0045            DSPROD(KI)=DSPROD(KI)-DCOR
  ISN 0046            WRITE(6,100) KI,DSPROD(KI)
  ISN 0047        100 FORMAT(' DSPROD(',I3,')= ',F13.4)
  ISN 0048          6 CONTINUE
  ISN 0049          5 CONTINUE
  ISN 0050            RETURN
  ISN 0051            END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=58,SIZE=0000K,
                           SOURCE,BCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF
ISN 0002            FUNCTION TEST(DMAX,K)
ISN 0003            IMPLICIT REAL*8(D)
ISN 0004            DOUBLE PRECISION DETAX
ISN 0005            DOUBLE PRECISION YORMA
ISN 0006            DOUBLE PRECISION DLOGM
ISN 0007            COMMON DGSUM(20,20),DXSUM(20),DSPROD(400),X(50,20),NV,NP,NG,
                   1(6100),NIG(20),NCO(20),NCR(20)
ISN 0008            DMAX=RMAX
ISN 0009            DRMAX=1.DO-DMAX**2
ISN 0010            DK=K
ISN 0011            DV=NV
ISN 0012            DV=(DV-1.DO)*.5DO
ISN 0013            DN=NP
ISN 0014            DNN=DN-DK*1.DO
ISN 0015            D9M=.5DO*BETAX(DRMAX,DV,.5DO)
ISN 0016            DKK=DK-1.DO
ISN 0017            DTEST=BETAX(D9M,DKK,DNN)
ISN 0018            TEST=DTEST
ISN 0019            RETURN
ISN 0020            END
```

```
         COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=58,SIZE=0000K,
                            SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,NOXREF
               C     BETAP REQUIRES FUNCTION SUBROUTINES BETAX,CLOGM, AND YORMX
ISN 0002                 DOUBLE PRECISION FUNCTION BETAP(DP,DA,DB)
ISN 0003                 IMPLICIT REAL*8(D)
ISN 0004                 DIMENSION DARG(4),DFUN(4)
ISN 0005                 NOUT = 6
ISN 0006                   DZERO=0.D0
ISN 0007                   DONE=1.D0
ISN 0008                 IF(DP-DONE)  95,91,20
ISN 0009              20 IF(DSQRT(DP)*(DP+DONE).LE.DZERO)  GO TO 95
ISN 0011                 IF(DB.EQ.DONE)  GO TO 90
ISN 0013                 IF (DA.EQ.DONE)  GO TO 92
ISN 0015                 DL = DZERO
ISN 0016                 DEP = DONE/3.D0
ISN 0017                 DLA = -DP
ISN 0018                 DLX = DONE - DP
ISN 0019                 JJ = 0
ISN 0020              89 JEND = 3
ISN 0021              88   JJ = JJ + 1
ISN 0022                 DO 80 J81=1,JEND
ISN 0023                 DAP = DBLE(DL)/2.D0
ISN 0024                 IF(DONE-DEXP(DAP)*(1.D0+DAP)) 1,1,2
ISN 0026               2 IF(DABS(DL-DX)/DP.LT.1.D-12).AND.(DL.GT.1.D-12)  GO TO 195
ISN 0028                 DO 81 I81=1,2
ISN 0029                 DARG(I) = DL + DX*DFLOAT(I)
ISN 0030                 DFUN(I) = BETAX(DARG(I),DP,DB) - DP
ISN 0031                 IF(DFUN(I).EQ.DZERO) GO TO DARG(I)
ISN 0033                 IF(DFUN(1)*DFUN(2)) 81,1,82
ISN 0034              82 DU = DARG(1)
ISN 0035                 DUX = DFUN(1)
ISN 0036                 GO TO 80
ISN 0038              81 DL = DARG(1)
ISN 0039                 DLA = DFUN(1)
ISN 0040                 GO TO 80
ISN 0041              81 CONTINUE
ISN 0042                 DL = DARG(2)
ISN 0043                 DLX = DFUN(2)
ISN 0044              80 CONTINUE
ISN 0045                 JEND = 2
ISN 0046                 DAP = DBLE(DL)/2.DO
ISN 0047                 DPS = DUX - DLX
ISN 0048                 IF(DABS(DFD).LT.1.D-10 ).AND.(DFD/DP.LT.1.D-6)  GO TO 1
ISN 0050                 DECH = DUX*(DL-DL)/DFD
ISN 0051                 DHP = DU - DECH
ISN 0052                 IF(DSIGN(DP - DL).LT.1.D-40)  GO TO 195
ISN 0054                 IF(DSIGN(DHP - DL).LT.1.D-12).AND.(DL.GT.1.D-12)  GO TO 195
ISN 0056                 DFUN(I) = BETAX(DHP,DA,DB) - DP
ISN 0057                 DABF = DABS(DFUN(3))
ISN 0058                 DFUN3    = DFUN(3)
ISN 0059                 IF(DABF.LT.1.D-10).AND.(DABF/DP.LT.1.D-6)  GO TO 1
ISN 0061                 IF (DHP.LT.1.D-40) GO TO 195
ISN 0063                 IF((DL-DHP).LT.1.D-12).AND.(DU.GT.0.99999999999D0)) GO TO 195
ISN 0065                 IF (DFUN(3)) 83,1,84
ISN 0066              83 IF(DZER.LE.C.5DO*(DU-DL)) GO TO 185
```

```
ISN 0088            DPPU # DMP
ISN 0089            DMP # 5.00*(DPP-DL) & DL
ISN 0070            DFUNE   # BETAX(DMP,DM,DN) - DP
ISN 0071            IF(DFUNE  < 183,1,40
ISN 0072       40   DU # DMP
ISN 0073            DUX # DFUNE
ISN 0074            DL # DMPU
ISN 0075            DLX # DFUN(3)
ISN 0076            IF(JJ-10) 88,89,195
ISN 0077       84   IF(DECR.GE.0.100*DU-DL)  GO TO 184
ISN 0079            DMPU = DMP
ISN 0080            DMP = DU - 5.00*DEC.
ISN 0081            DFUNE   # BETAX(DMP,DM,DN) - DP
ISN 0082            IF(DFUNE  <  41,1,184
ISN 0083       41   DU # DMPU
ISN 0084            DUX # DFUN(3)
ISN 0085            DL # DMP
ISN 0086            DLX # DFUNE
ISN 0087            IF(JJ-10) 88,89,195
ISN 0088       183  DL # DMP
ISN 0089            DLX # DFUNE
ISN 0090            IF(JJ-10) 88,89,195
ISN 0091       184  DL # DMP
ISN 0092            DLX # DFUNE
ISN 0093            IF(JJ-10) 88,89,195
ISN 0094       1    BETAP # DMP
ISN 0095            RETURN
ISN 0096       195  DRES # DFUNE   & DP
ISN 0097            WRITE(NOUT,196) DP,DM,DN,DMP,DRES
ISN 0098       196  FORMAT(1H0,5X,43HNO CONVERGENCE IN BETAP IN DOUBLE PRECISION /
                    11X,9HINPUT P =  D16.10,4H M =  D18.10,4H N = D18.10,9H LAST X =
                    2D16.10,13H PRODUCES P = D18.10)
ISN 0099            GO TO 1
ISN 0100       91   DMP # DP
ISN 0101            GO TO 1
ISN 0102       55   DMP # DZ
ISN 0103            WRITE (NOUT,101) DP,DM,DN
ISN 0104       101  FORMAT (1H0,28HARGUMENTS FOR BETAP WERE P =  D12.4, 4H M =  D12.4,
                    14H N =  D12.4, 28H RESULT HAS BEEN SET TO ZERO )
ISN 0105            GO TO 1
ISN 0106       90   DMP # DU - (DL-DP)**(DU/DN)
ISN 0107            GO TO 1
ISN 0108       92   DMP = DP**(DU/DN)
ISN 0109            GO TO 1
ISN 0110             END
```

```
         COMPILER OPTIONS - NAME= MAIN,OPT=00,LINECNT=56,SIZE=0000K,
                            SOURCE,BCD,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOID,NOXREF
ISN 0002            DOUBLE PRECISION FUNCTION YORMX(DZ)
ISN 0003            IMPLICIT  REAL*8(D)
ISN 0004            DOUBLE PRECISION YORMX
ISN 0005            DPI = .39894228040C1433
ISN 0006            DX = DABS(DZ)
ISN 0007            IF(DX.GT.3.D0) GO TO 10
ISN 0008            DAL = 0.D0
ISN 0009            DBL = 1.D0
ISN 0010            DAM = DX
ISN 0011            DBM = 1.D0
ISN 0012            DAN = 0.D0
ISN 0013            DBN = 0.D0
ISN 0014          5 DAN = DAN + 1.D0
ISN 0015            DAI = -(2.D0*DAN - 1.D0) *DX*DX
ISN 0016            DBI = 4.D0*DAN - 1.D0
ISN 0017            DAL = DBI*DAM + DAI*DAL
ISN 0018            DBL = DBI*DBM + DAI * DBL
ISN 0019            DAI = DX*DX - DAI
ISN 0020            DBI = 2.D0 + DBI
ISN 0021            DAM = DBI*DAL + DAI*DAM
ISN 0022            DBM = DBI*DBL + DAI*DBM
ISN 0023            DFA = DAL/DBL
ISN 0024            DFB = DAM/DBM
ISN 0025            IF(DFB.EQ.0.D0) GO TO 20
ISN 0027            IF(DABS((DFB-DFA)/DFA).LE.1.D-14) GO TO 20
ISN 0029            GO TO 5
ISN 0030         10 DAL = 0.D0
ISN 0031            DBL = 1.D0
ISN 0032            DAM = 1.D0
ISN 0033            DBM = DX
ISN 0034            DBI = DX
ISN 0035            DAN = 1.D0
ISN 0036            DFA = 1.D0/DX
ISN 0037         15 DAN = DAN + 1.D0
ISN 0038            DBI = DAN - 1.D0
ISN 0039            DAC = DBI*DAM + DAI*DAL
ISN 0040            DBC = DBI*DBM + DAI*DBL
ISN 0041            DFB = DAC/DBC
ISN 0042            DAL = DAM
ISN 0043            DBL = DBM
ISN 0044            DAM = DAC
ISN 0045            DBM = DBC
ISN 0046            IF(DFB.EQ.0.D0) GO TO 20
ISN 0048            IF(DABS((DFB-DFA)/DFB).LE.1.D-14) GO TO 20
ISN 0050            DFA = DFB
ISN 0051            GO TO 15
ISN 0052         20 YORMX = DPI*DFB*DEXP(-DX*DX/2.D0)
ISN 0053            IF(DX.LE.3.D0) YORMX = 0.500 - YORMX
ISN 0055            IF(DZ.GT.0.D0) YORMX = 1.D0 - YORMX
ISN 0057            RETURN
ISN 0058            END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=53,SIZE=0000K,
                           SOURCE,BCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF
ISN 0002                DOUBLE PRECISION FUNCTION DLGGM(DX)
ISN 0003                IMPLICIT REAL *8(D)
ISN 0004                DY=DX
ISN 0005                CTERM=1.D0
ISN 0006                IF(DX)1,1,2
ISN 0007      1         DLGGM=0.D0
ISN 0008                RETURN
ISN 0009      2         IF(DY-12.D0) 3,3,4
ISN 0010      3         CTERM=CTERM*DY
ISN 0011                DY=DY+1.D0
ISN 0012                GOTO 2
ISN 0013      4         DLGGM= (DY-.5D0)* DLOG(DY)-DY+1.D0/(12.D0*DY)-1.D0/(360.D0*DY**3)
                       1+ 1.D0/ (1260.D0*DY**5) -1.D0/ (1680.D0*DY**7 )+.9189385332046730D
                       2  -DLOG(CTERM)
ISN 0014                RETURN
ISN 0015                END
```

```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=58,SIZE=0000K,
                          SOURCE,EBCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF
ISN 0002            DOUBLE PRECISION FUNCTION BETAX(DX,DM,DN)
ISN 0003            IMPLICIT REAL *8( C,V,B)
ISN 0004            DTEST=.5979999999999D0
ISN 0005            DZ=0.0D
ISN 0006            DE= 1.0D
ISN 0007          DX = DM - DE
ISN 0008            DSUM=DZ
ISN 0009            IF( DM.LE. DZ) GOTO1
ISN 0011            IF( DN.LE. DZ) GOTO2
ISN 0013            IF( DX*(DE-DX) .LT. DZ) GOTO1
ISN 0015            IF( DX.GT.DTEST) GOTO2
ISN 0017          IF (DX.LT.1.0D-42) GO TO 3
ISN 0019          DCUB=DMIN1(2000.D0,30.D0/(DX*(1.D0-DX)))
ISN 0020            IPLACE=0
ISN 0021          IF (DX.LE.DP/(DM+DN)) GO TO 101
ISN 0023            DX=DE-DX
ISN 0024            DPLE=DM
ISN 0025            DM=DP
ISN 0026            DN=DPLE
ISN 0027          DX = DM + DE
ISN 0028            IPLACE=1
ISN 0029     101    IF(DM-DE. .5D0) GOTO7
ISN 0031          IF(DM+DN.GE.DCUB)GO TO 7C0
ISN 0033            IF(DN*.DDC
ISN 0034            ENT=1
ISN 0035            IF( DABS(DX-DNT) .LT. 1.D-7) GOTO4
ISN 0037            LI= 20.DD-DM*DE
ISN 0038            DN=DP-DE
ISN 0039            DLGN=DLOGM(DX)
ISN 0040            DNX=DN*DLOG( DE-DX)
ISN 0041          DLDX = DLOG(DN)
ISN 0042            DO 5 I=1,LI
ISN 0043            DX=DE*DE
ISN 0044          IF (I.GE.2) GO TO 305
ISN 0046            DT=DLOGM(DN+DX)- DLGN-(LOGM(DX+DE)+DX*DLOG(DX)+DNX
ISN 0047            IF(DT.GT. -6C.DD) GOTO6
ISN 0049          GO TO 5
ISN 0050      305   DT = DT + DLOG((DN + DX)/(DX + DE)) + DLDX
ISN 0051          IF (DT.GT.-60.DD) GO TO 6
ISN 0053     *5     CONTINUE
ISN 0054            GOTO7
ISN 0055     6      DSUM=DSUM+EXP(DT)
ISN 0056            DTERM=DSUM
ISN 0057            LI= 20.DD-DX
ISN 0058            DO 9 I=1,LI
ISN 0059            DTERM=( DN*DF)*DX*DTERM/(DX+DE)
ISN 0060            IF(DTERM/DSUM.LT. 1.D-12) GOTO8
ISN 0062            DSUM=DSUM+DTERM
ISN 0063     9      DN=DN+DE
ISN 0064         7 IF(DX+DM.GT.DCUB)GO TO 7C0
ISN 0066            DCUT = DZ
ISN 0067            DSML = DSUM
ISN 0068            DX = DX + DE
```

```
ISN 0068          DSN = DE
ISN 0069          CAL = C2
ISN 0070          CBL = DE
ISN 0071          CAH = DE
ISN 0072          CBH = DE
ISN 0073          DCFLT = DLOGM(DK*DN) - DLOGM(DK*DE) - DLOGM(DN) + DK*DLOG(DX) +
                 1DN * DLOG(DE - DX)
ISN 0074          DCFLU = DCFLT
ISN 0075          KFLAG = 1
ISN 0076          DSUM = DE
ISN 0077          KFLAG = 1
ISN 0078          DO 135 KK = 1,80
ISN 0079          DAK = KK
ISN 0080          DC1 = DK + DAK - DE
ISN 0081          DC2 = DK + 2.DO*DAK - DE
ISN 0082          DCN1 = -DK*DC1*(DC1+DN)/(DC2*(DC2-DE))
ISN 0083          DCN2 = DK*DAK*(DN-DAK)/(DC2*(DC2+DE))
ISN 0084          DAL = DAH * DCN1*DAL
ISN 0085          DAH = DAL + DCN2*DAH
ISN 0086          DBL = DBH * DCN1*DBL
ISN 0087          DBH = DBL + DCN2*DBH
ISN 0088          IF(DABS(DBH).LT.1.D-60) GO TO 201
ISN 0089          IF((DCFLU.LT.-80.D0).OR.(DCFLU.GT.80.D0)) GO TO 201
ISN 0090          IF(KFLAG.EQ.0) GO TO 203
ISN 0091          DCFT = DEXP(DCFLT)*DSM
ISN 0092          KFLAG = 0
ISN 0093          DSUM = DCFT
ISN 0094          KFLAG = 0
ISN 0095    203   DCFU = DSUM
ISN 0096          DSUM = DCFT*DAH/DBH
ISN 0097          IF(DCN2.EQ.DZ) GO TO 208
ISN 0098          IF (DSUM) 135,208,202
ISN 0099    202   IF(DABS(DSUM-DCFU)/DSUM.LT.1.D-12) GO TO 208
ISN 0100          GO TO 135
ISN 0101    201   IF(DSM.EQ.DZ) GO TO 135
ISN 0102          KFLAG = 1
ISN 0103          DCFLU = DCFLT + DLOG(DABS(DAH)) - DLOG(DABS(DBH)) + DLOG(DSUM)
ISN 0104          DSM = DSIGN(DE,DAH)*DSIGN(DE,DBH)*DSM*DSIGN(DE,DSUM)
ISN 0105    135   CONTINUE
ISN 0106    208   IF(KFLAG.EQ.1) DSUM = D2
ISN 0107          DSUM = DSUM + DSML
ISN 0108          GO TO 6
ISN 0109    700   DTH=DE/3.DO
ISN 0110          DK=DK+DE
ISN 0111          DN1=(DN+DK)**DTH
ISN 0112          DN2= (DK*(DE-DX))**DTH
ISN 0113          DU= 3.DO*(DN1*(DE-DE/(9.DO*DN))-DN2*(DE-DE/(9.DO*DK)))/
                 1  DSQRT( DN1*DN1/DN+DN2*DN2/DK)
ISN 0114          DSUM=DSUM+YORMX(DU)
ISN 0115    6     BETAX=DSUM
ISN 0116          IF (KFLAG.EQ. 0) RETURN
ISN 0117          BETAX=DE-BETAX
ISN 0118          DX=DE-DX
ISN 0119          DN=DN
ISN 0120          DN=DFLO
ISN 0121          RETURN
```

```
ISN 0131          4     DN = DNT
ISN 0132                GO TO 7
ISN 0133          1     WRITE(6,100) CN,CN,CX
ISN 0134        100     FORMAT(///5X, 46HERROR IN INPUT PARAMETER BETAX SET TO 0. ,
                       1 /5X, 2HN=, D20.8,2HN=,D20.8,2HX=,D20.8)
ISN 0135          3     BETAX=CZ
ISN 0136                RETURN
ISN 0137          2     BETAX=CE
ISN 0138                RETURN
ISN 0139                END
```

APPENDIX B

Source Listings for ELLIPSE Program

```
FORTRAN IV G LEVEL 19          MAIN          DATE = 71257          14/31/15

C THIS IS THE MAIN PROGRAM
0001        COMMON GDSX,GDSY,GTEXT,GPOINT,GDSE,GDSER,X(50,8),NV,NP,NGG,NSOL,
           1RC,CLREAN(5,8),A(36,11),XMAX(8),XMIN(8),XLO,YLO,XHI,YHI,NDET(5),
           2R(16),RWRKNG(16),GINPUT,RINPUT(8),SUC(10,8),INNSOL(50,2),IGWORK(50
           3),NIG(5),DSPROD(36),IG(50),DOSUM(5,8),DXSUM(8),AWRKNG(40)
0002        CALL CALC
0003        CALL EXIBIT
0004        STOP
0005        END
```

```
FORTRAN IV G LEVEL  19                    KD                    DATE = 71257          14/31/24

0001                FUNCTION KD(I,J)
0002                IF(I.LT.J) GO TO 1
0003                KD=(I*(I-1))/2+J
0004                GO TO 99
0005              1 KD=(J*(J-1))/2+I
0006             99 RETURN
0007                END
```

```
FORTRAN IV G LEVEL 19                CALC                DATE = 71257          14/31/49

0001              SUBROUTINE CALC
0002              COMMON GDSX,GDSY,GTEXT,GPOINT,GOSE,GOSER,X(50,9),NV,NP,NGG,NSOL,
                 1NG,CLMEAN(5,8),A(36,11),X*AX(9),XMIN(8),XLO,YLO,XHI,YHI,NDET(5),
                 2R(15),ARRANG(16),GINPUT,GINPUT(8),SUG(10,8),INNSOL(50,2),IGWORK(50
                 3),NIG(5),DSPROD(36),IG(50),DGSUM(5,9),DXSUM(8),ARRANG(40)
0003              READ (5,2) NP,NV,NG,NSOL
0004              NVV = (NV*(NV+1))/2
0005            2 FORMAT(4(1X,I3))
0006              DO 14 I=1,NG
0007           14 READ (5,451) (CLMEAN(I,J),J=1,NV)
0008          451 FORMAT (14X,6F7.3)
0009              DO 114 I=1,NSOL
0010          114 READ (5,451) (SUG(I,J),J=1,NV)
0011         1650 FORMAT(1X,12,11,110,8F7.3)
0012              DO 10 I=1,NP
0013              READ (5,1650) INNSOL(I,2),IG(I),INNSOL(I,1),(X(I,J),J=1,NV)
0014           10 CONTINUE
0015           13 DO 425 I=1,NG
0016          425 NIG(I) = 0
0017              DO 422 I=1,NG
0018              DO 423 J=1,NP
0019              IF (IG(J).EQ.I) NIG(I) = NIG(I) + 1
0020          423 CONTINUE
0021          422 CONTINUE
0022              NSUM = 0
0023              DO 429 I=1,NG
0024          429 NSUM = NSUM + NIG(I)
0025              DO 428 I=1,NP
0026              IGWORK(I) = IG(I)
0027          428 CONTINUE
0028              NGG = NG
0029              CALL CORZ
0030              DO 434 I=1,NVV
0031          434 DSPROD(I) = DSPROD(I)/(NMIC-NG)
0032              CALL SINV(DSPROD,NV,1.E-7,IER)
0033              WRITE (6,1661) IER
0034         1661 FORMAT (1X,'ERROR CODE AFTER CALL TO SINV IS',I3)
0035              DO 435 I=1,NVV
0036              A(I,1) = DSPROD(I)
0037          435 CONTINUE
0038              IF (INSCL.EQ.0) GO TO 450
0039              DO 455 I=1,NV
0040              DXSUM(I) = 0.0
0041              DO 456 J=1,NP
0042          456 DXSUM(I) = DXSUM(I) + X(J,I)
0043              DXSUM(I) = DXSUM(I)/NP
0044          455 CONTINUE
0045              DO 426 I=1,NSOL
0046              DO 427 J=1,NP
0047              IGWORK(J) = 0
0048              ICX = INNSOL(J,1)/10**(10-I)
0049              ICY = ICX/10
0050              IF (10*ICY.NE.ICX) IGWORK(J) = IG(J)
0051          427 CONTINUE
0052              DO 432 K=1,NG
0053          432 NIG(K) = 0
```

```
0054            DO 430 K=1,NG
0055            DO 431 J=1,NP
0056            IF (IGNORK(J) .EQ. K) NIG(K) = NIG(K) + 1
0057          431 CONTINUE
0058          430 CONTINUE
0059            NNIG = 0
0060            DO 441 K=1,NG
0061          441 NNIG = NNIG + NIG(K)
0062            CALL CORIS
0063            DO 442 K=1,NVV
0064          442 DSPROD(K) = DSPROD(K)/NNIG
0065            CALL SINV(DSPROD,NV,1.E-7,IER)
0066            WRITE (6,1661) IER
0067            DO 433 J=1,NVV
0068            A(J,I+1) = DSPROD(J)
0069          433 CONTINUE
0070          426 CONTINUE
0071          450 DO 15 I=1,NV
0072            XMAX(I) = X(1,I)
0073            XMIN(I) = X(1,I)
0074            DO 15 J=1,NP
0075            IF (X(J,I).GT.XMAX(I)) XMAX(I) = X(J,I)
0076            IF (X(J,I).LT.XMIN(I)) XMIN(I)=X(J,I)
0077           15 CONTINUE
0078            WRITE (6,1651) XMAX(1)
0079         1651 FORMAT(1X,'XMAX(1)=',F7.3)
0080            RETURN
0081            END
```

```
FORTRAN IV G LEVEL  19                  COR1S                   DATE = 71257          14/32/03

0001                  SUBROUTINE COR1S
0002                  COMMON GDSX,GDSY,GTEXT,GPOINT,GDSE,GDSER,X(50,8),NV,NP,NGG,NSOL,
                     1NG,CLMEAN(5,8),A(36,11),XMAX(8),XMIN(8),XLO,YLO,XHI,YHI,NDET(5),
                     2R(16),RWRKNG(16),GINPUT,RINPUT(8),SUG(10,8),INNSCL(50,2),IGWORK(50
                     3),NIG(5),DSPROD(36),IG(50),DGSUM(5,8),DXSUM(8),AWRKNG(40)
0003                  DO 10 I=1,NV
0004                  DO 11 J=1,I
0005                  KI = KB(I,J)
0006                  DSPROD(KI) = 0.0
0007                  DO 12 L=1,NP
0008                  IF (IGWORK(L).EQ.0) GO TO 12
0009                  IF (IG(L).EQ.0) GO TO 7
0010                  K = IG(L)
0011                  DX1 = X(L,J) - CLMEAN(K,J)
0012                  DX2 = X(L,I) - CLMEAN(K,I)
0013                  DSPROD(KI) = DSPROD(KI) + DX1*DX2
0014                  GO TO 12
0015                7 DX1 = X(L,J) - DXSUM(J)
0016                  DX2 = X(L,I) - DXSUM(I)
0017                  DSPROD(KI) = DSPROD(KI) + DX1*DX2
0018               12 CONTINUE
0019                  WRITE (6,100) KI,DSPROD(KI)
0020              100 FORMAT(' DSPROD(',I3,')= ',F13.4)
0021               11 CONTINUE
0022               10 CONTINUE
0023                  RETURN
0024                  END
```

```
FORTRAN IV G LEVEL 19                COR2              DATE = 71257        14/32/11

0001            SUBROUTINE COR2
0002            COMMON GDSX,GDSY,GTEXT,SPOINT,GDSE,GDSER,X(50,8),NV,NP,NGG,NSOL,
               1NG,CLMEAN(5,8),A(36,11),XMAX(8),XMIN(8),XLO,YLO,XHI,YHI,NDET(5),
               2R(10),RANKNG(16),GINPUT,RINPUT(8),SUG(10,8),INNSCL(50,2),IGWORK(50
               3),NIG(5),DSPROD(36),IG(50),DGSUM(5,8),DXSUM(8),ARRANG(40)
0003            NC=(NV*(NV+1))/2
0004            DO 2 I=1,NC
0005          2 DSPROD(I)=0.00
0006            DO 11 I=1,NGG
0007            DO 12 J=1,NV
0008         12 DGSUM(I,J)=0.00
0009         11 CONTINUE
0010            DO 13 K=1,NGG
0011            DO 3 I=1,NV
0012            DO 4 J=1,NP
0013            DX=X(J,I)
0014            IF(IGWORK(J).NE.K) GO TO 4
0015            DGSUM(K,I)=DGSUM(K,I)+DX
0016          4 CONTINUE
0017          3 CONTINUE
0018         13 CONTINUE
0019            DO 57 I=1,NV
0020            DO 58 K=1,NGG
0021            WRITE(6,159) I,K,DGSUM(K,I)
0022        159 FORMAT(' SUM OF CASES FOR VAR',I2,' IN GROUP',I2,'= ',D12.4)
0023         58 CONTINUE
0024         57 CONTINUE
0025            WRITE(6,300)
0026        300 FORMAT(' DSPROD IS UPPER TRI PART OF MATRIX OF CORRECTED S.S. AND
              1S.P. FOR NV VARS, STORED BY COLS')
0027            DO 5 J=1,NV
0028            DO 6 I=1,J
0029            KI=KO(I,J)
0030            DO 7 L=1,NP
0031            IF(IG(L).NE.0) GO TO 7
0032            DX1=X(L,J)
0033            DX2=X(L,I)
0034            DSPROD(KI)=DSPROD(KI)+DX1*DX2
0035          7 CONTINUE
0036            DCOR=0.00
0037            DO 14 L=1,NGG
0038            IF(NIG(L).EQ.0) GO TO 23
0039            DCOR=DCOR+DGSUM(L,J)*DGSUM(L,I)/NIG(L)
0040            GO TO 14
0041         23 NGG = NGG-1
0042         14 CONTINUE
0043            DSPROD(KI)=DSPROD(KI)-DCOR
0044            WRITE(6,100) KI,DSPROD(KI)
0045        100 FORMAT(' DSPROD(',I3,')= ',F13.4)
0046          6 CONTINUE
0047          5 CONTINUE
0048            RETURN
0049            END
```

```
FORTRAN IV G LEVEL 19                EXIBIT          DATE = 71257        14/32/21

0001              SUBROUTINE EXIBIT
0002              COMMON GDSX,GDSY,GTEXT,GPOINT,GDSE,GDSER,X(50,8),NV,NP,KGG,NSOL,
                 1NG,CLMEANS(8),A(36,11),XMAX(8),XMIN(8),XLO,YLO,XHI,YHI,NDET(5),
                 2R(16),RWRKNG(16),GINPUT,GINPUT(8),SUG(10,3),IKNSOL(50,2),IGWORK(50
                 3),NIG(5),DSPROD(36),IG(5),DSSUM(5,8),DKSUM(8),ARROXG(60)
0003              CALL DISPLA(GDSX,GDSY,GTEXT,GPOINT,GDSE,GDSER,GINPUT)
0004              WRITE (6,1654)
0005           1654 FORMAT(1X,'DISPLA SUCCESSFUL')
0006              I = LIGHTS(1,1,-NV,27,-2,,1)
0007              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0008              CALL MESSGE
0009              WRITE (4,1655)
0010           1655 FORMAT('1FOLLOWING ANY DISPLAY OF PROJECTIONS PRESS'/
                 1'PFKEY 29 TO CHANGE THE VECTOR FORMING ONE OF THE AXES'/
                 2'PFKEY 31 TO CHANGE THE VARIABLE FORMING ANOTHER AXIS.'/
                 3'PFKEY PRESS ANY KEY TO GO ON.')
0011              CALL ERMTS(GTEXT)
0012              LP = PLOT(GTEXT)
0013              LP = OBTAIN(NDET)
0014              LP = UNPLOT(GTEXT)
0015              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0016              NTEST = 1
0017              NGG = 0
0018           20 CALL KEYIN (NAXIS,NTEST)
0019              IF (NAXIS.EQ.30)GO TO 40
0020              CALL ECLPSE (NAXIS)
0021           10 KK = OBTAIN(NDET)
0022              KK = NDET(4)
0023           17 IF (KK.EQ.30) GO TO 40
0024              NTEST = KK
0025              IF (KK.EQ.29) GO TO 60
0026              IF (KK.EQ.31) GO TO 80
0027              IF (KK.LE.NSOL) GO TO 61
0028              LP = UNPLOT(GTEXT)
0029              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0030              WRITE (4,62)
0031           62 FORMAT('1CARDAN OR SINGULAR SOLUTION - PRESS ANOTHER KEY'/
                 1'2 IF TRAPPED,PRESS KEY 29')
0032              CALL ERMTS(GTEXT)
0033              LP = PLOT(GTEXT)
0034              GO TO 10
0035           80 LP = UNPLOT(GDSX,GDSY,GPOINT,GTEXT)
0036              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0037              LP = UNPLOT(GDSE)
0038              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0039              GO TO 20
0040           61 CALL RELPSE (KK,NAXIS)
0041              IF (KK.NE.-15) GO TO 103
0042              KK = 15
0043              GO TO 17
0044          103 KK = OBTAIN(NDET)
0045              KK = NDET(4)
0046              LP = UNPLOT(GDSER)
0047              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0048              GO TO 17
0049           40 CALL BLANK
```

```
0050            CALL RESETI(GDSX,GDSY,GDSE,GDSER,GTEXT,GPOINT,GINPUT)
0051            RETURN
0052            END
```

```
FORTRAN IV G LEVEL  19                MESSGE              DATE = 71257          14/32/29

0001                  SUBROUTINE MESSGE
0002                  COMMON GDSX,GDSY,GTEXT,GPOINT,GDSE,GDSER,X(50,8),NV,NP,NGG,NSOL,
                     1NG,CLMEAN(5,8),A(36,11),XMAX(8),XMIN(8),XLO,YLO,XHI,YHI,NDET(5),
                     2R(16),AWRKNG(16),GINPUT,RINPUT(8),SUG(10,8),INNSCL(50,2),IGWORK(50
                     3),NIG(5),DSPROD(36),IG(50),DGSUM(5,8),DXSUM(8),AWRKNG(40)
0003                  WRITE (4,460)
0004                460 FORMAT('1THIS PROGRAM DISPLAYS CLUSTERS BY PROJECTING THEM ON'/
                     1'0VARIOUS 2-DIMENSIONAL SUBSPACES.SIMPLE STRUCTURE'/
                     2'0SOLUTIONS CAN ALSO BE INDICATED.REFER TO YOUR'/
                     3'0INSTRUCTION CHART.HAVE YOU ENTERED ALL NECESSARY'/
                     4'0DATA VIA IEBGENER?'/
                     5'0',10X,'THE BOTTOM ROW OF THE PROGRAM '/
                     6'0FUNCTION KEYS IS LIT UP.THEY WILL BE USED'/
                     7'0AS DIRECTED.THE DARK ONE,KEY NO. 30,IS'/
                     8'0THE PANIC BUTTON.IT WILL RETURN YOU TO THE '/
                     9'0MONITOR.'/'0IN CASE OF PANIC PRESS KEY 30'/
                     1'0NOW PRESS ANY KEY TO GET STARTED')
0005                  CALL XKFMTS(GTEXT)
0006                  LP = PLOT(GTEXT)
0007                  WRITE (6,1653) LP
0008               1653 FORMAT(1X,'LP AFTER FIRST MESSAGE IS',I4)
0009                  I = DETAIN(NDET)
0010                  I = UNPLOT(GTEXT)
0011                  CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0012                  RETURN
0013                  END
```

```
FORTRAN IV G LEVEL 19                KEVIN                DATE = 71257          14/32/38

0001            SUBROUTINE KEVIN (MAX S,NTEST)
0002            COMMON GDSX,GDSY,GTEX ,GPOINT,GOSE,GDSER,X(50,6),NV,N*,NGG,NSOL,
               1NG,CLMEAN(5,8),A(36,1 1),XMAX(9),XMIN(3),XLO,YLO,XHI,YHI,NDET(5),
               2R(16),RANKNG(16),GINPUT,RINPUT(8),SUG(10,8),INNSOL(50,2),IGNORE(50
               3),NIG(5),DSPROD(36),I(50),DGSUM(5,8),DKSUM(8),AVRANG(40)
0003            IF (NTEST.EQ.31) GO T) 101
0004          99 CALL BLANK
0005            CALL RCURS
0006            CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GOSE,GDSER,GINPUT)
0007            WRITE (4,193) NSOL
0008         190 FORMAT('PL'/'PO   ACCORDING TO THE ROTATION PROGRAM THE'/
               1'P FOLLOWING VECTORS TRANSFORM RAW DATA INTO'/
               2'P SIMPLE STRUCTURE SOLUTIONS NO.1 TO',I3)
0009            CALL WRFNTS(GINPUT)
0010            DO 110 J=1,NSOL
0011            WRITE (4,191) (SUG(J,K),K=1,NV)
0012         191 FORMAT('P  ',8F8.3)
0013            CALL WRFNTS(GINPUT)
0014            IDUM = PLOT(GINPUT)
0015            CALL RESET(GINPUT,GTEXT)
0016         110 CONTINUE
0017            WRITE (4,3000)
0018        3000 FORMAT('PENTER A TRANSFORMATION VECTOR.'/
               1'P NO MORE THAN 7 CHARACTERS.DECIMAL POINT'/
               2'P MUST BE TYPED'/
               3'P DEPRESS JUMP KEY AFTER EACH COORDINATE'/
               5'P X(1)='/'U.',10X/
               6'P X(2)='/'U.',10X/
               7'P X(3)='/'U.',10X/
               8'P X(4)='/'U.',10X/
               9'P X(5)='/'U.',10X/
               1'P X(6)='/'U.',10X/
               2'P X(7)='/'U.',10X/
               3'P X(8)='/'U.',10X/
               4'P IF PREVIOUS VECTOR OK,JUST PRESS EOB')
0019            CALL WRFNTS(GTEXT)
0020            J = PLOT(GTEXT)
0021            CALL CURSS(GTEXT,126)
0022            L = DETAIN (RSLT)
0023            CALL SCTOV(GTEXT)
0024            CALL DVTOM(GTEXT)
0025            DO 211 JJ = 1,NV
0026         211 RANKNG(JJ) = RINPUT(JJ)
0027            READ (4,3005) (RINPUT(I),I=1,NV)
0028        3005 FORMAT( /////G10.4//E10.4//G10.4//G10.4//G10.4//G10.4//G10.
               14)
0029            DO 213 IJ = 1,NV
0030            IF (RINPUT(IJ).NE.0.0) GO TO 214
0031         213 CONTINUE
0032            IF (NGG.EQ.0) GO TO 214
0033            DO 216 IJ = 1,NV
0034         216 RINPUT(IJ) = RANKNG(IJ)
0035         214 CALL BUFRS
0036            LO = UNPLOT(GINPUT)
0037            LP = UNPLOT(GTEXT)
0038            CALL BLANK
```

```
0039              NGG = 1
0040              CALL RCURS
0041              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0042            101 WRITE (4,1658)
0043           1658 FORMAT('P1 PRESS ONE KEY CORRESPONDING TO THE AXIS'/
                   1'P - VARIABLE YOU WISH TO SELECT, OR 30 IF'/
                   2'P  YOU WISH TO STOP.IF YOU WISH TO MAKE '/
                   3'P CHANGES IN YOUR VECTOR PRESS KEY 28.')
0044              CALL WRFMT$ (GTEXT)
0045              LP = PLOT(GTEXT)
0046             60 NAXIS = DETAIN(NDET)
0047              NAXIS = NDET(4)
0048              IF (NAXIS.EQ.30) RETURN
0049              IF (NAXIS.EQ.28) GO TO 99
0050              IF (NAXIS.LE.NV) GO TO 54
0051              LP = UNPLOT(GTEXT)
0052              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0053              WRITE(4,52)
0054             52 FORMAT('P1 ERROR - PRESS PROPER KEY')
0055              CALL WRFMT$(GTEXT)
0056              LP = PLOT(GTEXT)
0057              GO TO 60
0058             54 LP = UNPLOT(GTEXT)
0059              CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0060              RETURN
0061              END
```

```
FORTRAN IV G LEVEL 19                ELLIPSE          DATE = 71257          14/32/48

0001              SUBROUTINE ELLPSE (NAXIS)
0002              DIMENSION ELPSX(5,72),ELPSY(5,72),UE(400),VE(100)
0003              COMMON GDSX,GDSY,GTEXT,IPOINT,GDSE,GDSEN,X(50,8),NV,NP,NCG,NSOL,
                 1NG,CLMEAN(5,8),A(36,11),XMAX(8),XMIN(8),XLO,YLO,XHI,YHI,NDET(5),
                 2R(16),RRANG(16),GINPUT,RINPUT(8),SUG(10,8),INSGL(50,2),IGNORE(50
                 3),A1(5),DSPROD(36),(C(:,0),DGSUM(5,8),DXSUM(8),ARRAG(40)
0004              EQUIVALENCE (IGNORE(1),ELPSX(1,1),VE(1)),(ELPSY(1,1),UE(1))
0005              DO 10 K=1,NV
0006           10 R(K) = 0.0
0007              R(NAXIS) = 1.0
0008              NVV = NV*2
0009              KK = NV + 1
0010              DO 11 K=KK,NVV
0011           11 R(K) = RINPUT(K-NV)
0012              R(NV*NAXIS) = 0.0
0013              SS = 0.0
0014              DO 20 K= KK,NVV
0015              SS = SS + R(K)**2
0016           20 CONTINUE
0017              IF (SS.GT.0.0) GO TO 211
0018          213 WRITE (6,1659)
0019         1659 FORMAT('0',34X,'INCONSISTENT PRESS KEY 29 AND NEW VECTOR')
0020              CALL SKIPIS (GTEXT)
0021              LP = PLOT(GTEXT)
0022              RETURN
0023          211 DO 21 K=KK,NVV
0024              R(K) = R(K)/SQRT(SS)
0025           21 CONTINUE
0026              NVV = (NV*(NV+1))/2
0027              DO 12 K=1,NVV
0028           12 ARRANG(K) = A(K,1)
0029              CALL XPRD(ARRANG,R,R,KPG,NV,NV,1,0,2)
0030              CALL GIPRD(K,RRANG,ARRANG,NV,2,2)
0031              SII = ARRANG(1)
0032              SIJ = ARRANG(2)
0033              SJJ = ARRANG(4)
0034              DO 13 K=1,NV
0035              DO 13 KK = 1,NP
0036           13 U((K-1)*NP+KK) = X(KK,K)
0037              CALL GMPRO (U,R,V,NP,NV,2)
0038              DO 14 K=1,NP
0039              U(K) = V(K)
0040           14 V(K) = V(NP+K)
0041              VMAX = V(1)
0042              VMIN = V(1)
0043              DO 15 K=2,NP
0044              IF (V(K).GT.VMAX) VMAX = V(K)
0045              IF (V(K).LT.VMIN) VMIN = V(K)
0046           15 CONTINUE
0047              XLO = (-0.2*XMAX(NAXIS)+1.6*XMIN(NAXIS))/1.6
0048              XHI = (1.6*XMAX(NAXIS)-0.2*XMIN(NAXIS))/1.6
0049              YLO = (-0.2*VMAX+1.8*VMIN)/1.6
0050              YHI = (1.8*VMAX-0.2*VMIN)/1.6
0051              CALL UCORD (XLO,YLO,XHI,YHI)
0052              CALL PLACE(GDSX,XMIN(NAXIS),VMAX)
0053              CALL LINE(GDSX,XMIN(NAXIS),VMIN)
```

```
FORTRAN IV G LEVEL 19                ELLPSE              DATE = 71257        14/32/48

0054              CALL LINE(GDSX,XMAX(NAXIS),VMIN)
0055              LP = PLOT(GDSX)
0056              CALL PLACE(GDSY,XLO,0.5*(YMI+VMAX))
0057              WRITE (6,120) VMAX
0058          120 FORMAT(*B.*,F9.3)
0059              CALL WRFMTS(GDSY)
0060              CALL PLACE(GDSY,XLO,(2.*YLO+VMIN)/3.)
0061              WRITE (6,120) VMIN
0062              CALL WRFMTS(GDSY)
0063              XLX = (XLO+6.*AMIN(NAXIS))/7.
0064              YLX = (YLO+2.*VMIN)/3.
0065              CALL PLACE(GDSY,XLX,YLX)
0066              WRITE (6,121) AMIN(NAXIS)
0067          121 FORMAT(*B.*,F9.3)
0068              CALL WRFMTS(GDSY)
0069              XLXX = (9.*AMAX(NAXIS)-2.*XMI)/7.
0070              CALL PLACE(GDSY,XLXX,YLX)
0071              WRITE (6,120) XMAX(NAXIS)
0072              CALL WRFMTS(GDSY)
0073              LP = PLOT(GDSY)
0074              DO 110 K=1,NP
0075              CALL PLACE(GPOINT,U(K),V(K))
0076              WRITE (6,111) IXYSCL(K,2)
0077          111 FORMAT (*B.*,I2)
0078              CALL WRFMTS(GPOINT)
0079          110 CONTINUE
0080              LP = PLOT(GPOINT)
0081              ANGLE = 1.0
0082              DEN = (SII-SJJ)**2 + 4.*SIJ**2
0083              IF (DEN.NE.0.0) ANGLE = SQRT(0.5*SQRT(0.25-SII**2/DEN))
0084              AXIS1 = SII*ANGLE**2+SJJ*(1.-ANGLE**2)+2.*ANGLE*SQRT(1.-ANGLE**2)*
                 1SIJ
0085              IF (AXIS1.LE.0.0) GO TO 213
0086              AXIS1 = 1./SQRT(AXIS1)
0087              AXIS2 = (1. - ANGLE**2)*SII+ANGLE**2*SJJ
                 1-2.*ANGLE*SQRT(1.-ANGLE**2)*SIJ
0088              IF (AXIS2.LE.0.0) GO TO 213
0089              AXIS2 = 1./SQRT(AXIS2)
0090              SIG = SIGN(1.,(AXIS2-AXIS1)*SIJ)
0091              C = 3.141593/180.
0092              THETA = 5.0
0093              DO 31 K = 1,NV
0094              DO 31 KK = 1,NG
0095              XRANG ((K-1)*NG+KK) = CLMEAN (KK,K)
0096           31 CONTINUE
0097              CALL GMPRD (A,RANG,R,RERRANG,NG,NV,2)
0098              DO 150 K=1,NG
0099              DO 150 KK =1,72
0100              RADIAN = THETA*C*KK
0101              ELX = AXIS1*COS(RADIAN)
0102              ELY = AXIS2*SIN(RADIAN)
0103              ELPSX(K,KK) = RERRANG(K) + ELX*ANGLE - ELY*SQRT(1.-ANGLE**2)*SIG
0104              ELPSY(K,KK) = RERRANG(NG+K)+ELX*SQRT(1.-ANGLE**2)*SIG + ELY*ANGLE
0105          150 CONTINUE
0106              DO 100 K=1,NG
0107              DO 100 KK=1,72
```

```
0108              CALL POINT(GDSE,ELPSX(K,KK),ELPSY(K,KK))
0109          100 CONTINUE
0110              LP = PLOT(GDSE)
0111              KK = NV + 1
0112              NVV = NV*2
0113              WRITE (4,1657)NAXIS, (R(K),K=KK,NVV)
0114         1657 FORMAT('P1',10X,'VARIABLE',I2,'AGAINST VECTOR'/
                 2'P  ',8F8.3)
0115              CALL WRFMTS(STEXT)
0116              LP = PLOT(STEXT)
0117              RETURN
0118              END
```

```
FORTRAN IV G LEVEL 19          RELPSE          DATE = 71257          14/33/01

0001          SUBROUTINE RELPSE(KK,NA(15)
0002            DIMENSION RELPSX(5,72),RELPSY(5,72)
0003            COMMON GDSX,GDSY,GTEXT,GPOINT,GDSE,GDSER,X(50,9),NV,NP,NGG,KSOL,
               1NG,CLMEAN(5,8),A(36,11),AAA(6),XMIN(8),XLO,YLO,XHI,YHI,NOCT(5),
               2R(16),AAAANG(16),GINPUT,RINPUT(8),SUG(10,9),IANSCL(50,2),IGNO=K(50
               3),RIG(5),DSPRGO(36),IC(50),DCSUM(5,8),DXSUM(8),AAAANG(40)
0004            KX = NV + 1
0005            KY = 2*NV
0006            DO 212 JX = KX,KY
0007            IF(R(JX)**2.GT.0.0) GO TO 213
0008          212 CONTINUE
0009            GO TO 211
0010          213 CALL UCORD (XLO,YLO,XHI,YHI)
0011            NVV = (NV*(NV+1))/2
0012            DO 61 K=1,NVV
0013            AAAANG(K) = A(K,KK+1)
0014           61 CONTINUE
0015            CALL MPNDIA(AAANG,R,RAAANG,NV,NV,1,0,2)
0016            CALL GIPADIR,R,AAANG,AAAANG,NV,2,2)
0017            SII = AAAANG(1)
0018            SIJ = AAAANG(2)
0019            SJJ = AAAANG(4)
0020            ANGLE = 1.0
0021            DEN = (SII-SJJ)**2 + 4.*SIJ**2
0022            IF (DEN.NE.0.0) ANGLE = SQRT(0.5+SQRT(0.25-SIJ**2/DEN))
0023            AXIS1 = SII*ANGLE**2+SJJ*(1.-ANGLE**2)+2.*ANGLE*SQRT(1.-ANGLE**2)*
               1SIJ
0024            IF (AXIS1.LE.0.0) GO TO 211
0025            AXIS1 = 1./SQRT(AXIS1)
0026            AXIS2 = (1. - ANGLE**2)*SII+ANGLE**2*SJJ
               1-2.*ANGLE*SQRT(1.-ANGLE**2)*SIJ
0027            IF (AXIS2.LE.0.0) GO TO 211
0028            AXIS2 = 1./SQRT(AXIS2)
0029            SIG = SIGN(1.,(AXIS2-AXIS1)*SIJ)
0030            LP = UNPLOT(GTEXT)
0031            CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
0032            LL = NV + 1
0033            NVV = NV*2
0034            WRITE (6,1660) MAXIS,KK,(A(L),L=LL,NVV)
0035         1660 FORMAT('P1',2X,'VARIABLE',I2,'AGAINST VECTOR BELOW,(SIMPLE PLANE N
               10.',I2,'SUPERIMPOSED)'/'P   ',10X,8F6.3)
0036            CALL WRAPIT(GTEXT)
0037            LP = PLOT(GTEXT)
0038            C = 3.141593/180.
0039            THETA = 5.0
0040            DO 31 K = 1,NV
0041            DO 31 KK = 1,NG
0042            AAAANG I(K-1)*NG+KK) = CLMEAN (KK,K)
0043           31 CONTINUE
0044            CALL EXPND (AAAANG,R,RAAANG,NG,NV,2)
0045            DO 160 K=1,NG
0046            DO 160 KK=1,72
0047            RADIAN = THETA*C*KK
0048            ELX = AXIS1*COS(RADIAN)
0049            ELY = AXIS2*SIN(RADIAN)
0050            RELPSX(K,KK)=RAAANG(K)+ELX*ANGLE-ELY*SQRT(1.-ANGLE**2)*SIG
```

```
FORTRAN IV G LEVEL  19                    RELPSE                DATE = 71257          14/33/21

0051                  RELPSY(K,KK)=RWRKNG(NG+K)+ELX*SQRT(1.-ANGLE**2)*SIG+ELY*ANGLE
0052               160 CONTINUE
0053                  DO 170 K=1,NG
0054                  DO 170 KK=1,72
0055                  CALL POINT(GDSER,RELPSX(K,KK),RELPSY(K,KK))
0056               170 CONTINUE
0057                  LP = PLOT(GDSER)
0058                  RETURN
0059               211 KK = -15
0060                  RETURN
0061                  END
```