

AD735129



Reproduced by
**NATIONAL TECHNICAL
INFORMATION SERVICE**
Springfield, Va. 22151

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) University of Georgia		2a. REPORT SECURITY CLASSIFICATION 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 December 1971	7a. TOTAL NO. OF PAGES 156	7b. NO. OF REFS 38	
8a. CONTRACT OR GRANT NO. N00014-69-A-0423	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Report No. 75 Dept. of Statistics and Computer Science		
b. PROJECT NO. NR 042-261	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)		
c.	No. 17 THEMIS		
d.			
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 in-between 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 entity)		20. REPORT SECURITY CLASSIFICATION	
		20. 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.		8a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.			
c.		8b. 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.			
<p>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.</p> <p>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 choice of a particular plane of projection.</p>			

Unclassified

Security Classification

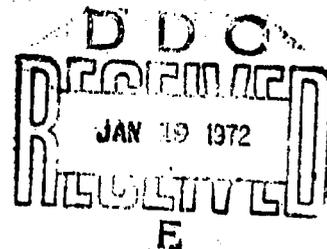
KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
CLASSIFICATION CLUSTER ANALYSIS FACTOR ANALYSIS MULTIVARIATE ANALYSIS STATISTICAL COMPUTATION CONVERSATIONAL COMPUTATION GRAPHICS TERMINALS						

THEMIS REPORT NUMBER 17
TECHNICAL REPORT NUMBER 75
CONFIGURATION AND CLASSIFICATION OF CLUSTERS IN
n-DIMENSIONS

SURENDRA J. TRIVEDI

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

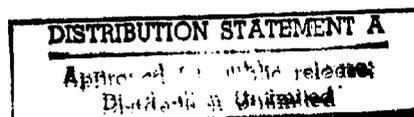


TABLE OF CONTENTS

CHAPTER		Page
I.	INTRODUCTION	1
	1.1 Multivariate Experiments	1
	1.2 Review of the Related Literature	2
	1.3 Real and Virtual Clusters	4
	1.4 d-clusters and k-clusters	5
	1.5 Identification of Mixed Samples	6
	1.6 Definition of the Problem	9
II.	PRINCIPAL COMPONENT AND OTHER PROJECTIONS	14
	2.1 Introduction	14
	2.2 Principal Component Projections	14
	2.3 Principal Component Displays	17
	2.4 Other Projections	18
III.	CLUSTERS IN SUBSPACES--THE SIMPLE STRUCTURE SUBSPACE .	19
	3.1 The Problem	19
	3.2 Mathematical Formulation	20
	3.3 A Basic Probability	21
	3.4 Determination of Simple Structure Configuration	26
IV.	CLUSTERS IN SUBSPACES--IDENTIFICATION	30
	4.1 Introduction	30
	4.2 Overdetermined Subspaces	30
	4.3 Identification of Subspaces	32
	4.4 Elimination of Variables	38

TABLE OF CONTENTS (continued)

CHAPTER	Page
V. DESCRIPTION OF COMPUTER PROGRAMS	43
5.1 The Computer Programs	43
5.2 An Algorithm for Identification of Points Lying on a Subspace	43
5.3 Loading the Data Set (Utility Program (IEBGENER)	46
5.4 The Display Program and the Conversational System	46
VI. USER'S GUIDE	68
6.1 Introduction	68
6.2 The CLUSTER Program	68
6.3 IEBGENER Utility Routine	70
6.4 The ELLIPSE Program	71
6.5 An Illustration	78
BIBLIOGRAPHY	114
APPENDIX	117
A	117
B	140

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 $n \times p$ 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 $N \times N$ 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 experimental 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_0 such that, under the null hypothesis of uniform distribution of points,

$$P [k_0 \text{ or more points in } S] < \alpha$$

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_0 be the radius of a sphere (or hypersphere) 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

$$P [d < d_0] < \alpha$$

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 $r \times s$ 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.

¹SSP - 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 independent, 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 variables 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 ($X : n \times m$), such that $XX' = F$ for m a minimum and, furthermore, satisfy all inequalities that whenever $r_{ij} > r_{kl}$ then $f_{ij} > f_{kl}$ 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 $(\text{length } OA)/(\text{length } OS)$. 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{y} .

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 sub-clusters--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

²Those 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 $\mu_1, \mu_2, \dots, \mu_k$ 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 $i = 1, 2, \dots, k$, 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 i th unit ellipsoid¹

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

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

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

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)

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

can be written as

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

and hence

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

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

$$(\underline{x} - \underline{\mu}_1)' Q D_{1/\lambda} Q' (\underline{x} - \underline{\mu}_1) = 1 \quad (2.2.5)$$

or, if we let $\underline{y}'_1 = (\underline{x} - \underline{\mu}_1)' Q$, this reduces to

$$\underline{y}'_1 D_{1/\lambda} \underline{y}_1 = 1 \quad (2.2.6)$$

(2.2.6) is the standard principal axes reduction of the conic (2.2.1). The transformation $\underline{y}'_1 = (\underline{x} - \underline{\mu}_1)' Q$ 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

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

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

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

Since Q is orthogonal, $\underline{z}'_1 Q$ will be coordinates of an original data point \underline{z}'_1 , which is the i th row of the data matrix Z , with reference to the new coordinate axes. In particular $\underline{z}'_1 \underline{q}_1, \underline{z}'_1 \underline{q}_2$ 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 $p \times p$ variance-covariance matrix, \underline{x} are the running coordinates and $\underline{\mu}_i$ is the cluster center of the i th 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

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

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 to

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

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 $p \times p$ 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

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

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

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

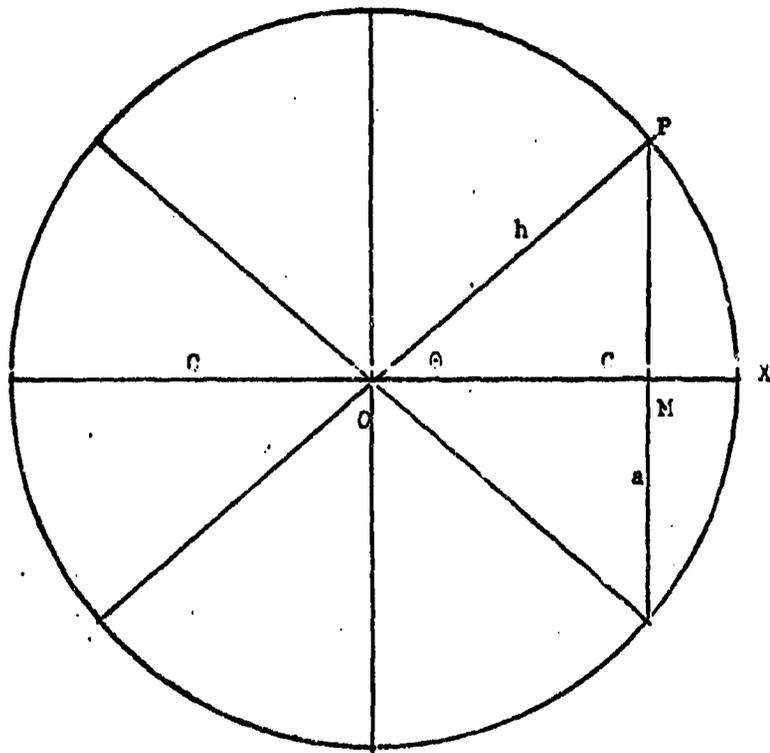


Figure 3.3.1

$$\begin{aligned}
 P_2 &= \frac{2 \times \text{length of arc generated by angle } 2\theta}{\text{circumference of the circle}} \\
 &= 2 \cdot OP \cdot 2\theta / 2\pi \cdot OP \\
 &= 2\theta / \pi \\
 &= (2/\pi) \arcsin(a/2h) \qquad (3.1)
 \end{aligned}$$

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 $ds = hdy/\sqrt{h^2 - y^2}$

$$\begin{aligned}
 \text{Hence } C_3 &= 2 \cdot 2\pi \int_0^{a/2} (h\sqrt{h^2 - y^2}) / (\sqrt{h^2 - y^2}) dy \\
 &= 2\pi ah
 \end{aligned}$$

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

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

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

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

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

$$\binom{n-p+1}{r-p+1} P_p^{r-p+1} (1-P_p)^{n-r}$$

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

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

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

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

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

$$\begin{aligned} &P[r \text{ or more of } n \text{ points lie within } \pm a/2] \\ &= B(P_p; r-p+1, n-r+1) \end{aligned} \quad (3.4.4)$$

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)

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

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.,

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

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 $n \times p$ 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 $n \times NG$ such that

$$\begin{aligned} a_{ij} &= 1 \quad \text{if unit } i \text{ belongs to cluster } j \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{4.3.1}$$

Let D_k (of order $NG \times NG$) denote a diagonal matrix with elements k_j , $j = 1, 2, \dots, NG$ 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,

$$Y = X - AD_k^{-1}A'X \quad (4.3.2)$$

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

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

$$S = (1/n_e)Y'Y \quad (4.3.4)$$

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,

$$S = TT' \quad (4.3.5)$$

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

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

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

$$Z\underline{t} = \underline{v} \quad (4.3.7)$$

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 i th row of Z and v_i denote the i th element of \underline{v} . Then it is clear that

$$\underline{z}_i' \underline{t} = v_i \quad (4.3.8)$$

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

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

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 $\underline{z}'_1/\sqrt{\underline{z}'_1\underline{z}'_1}$. Their "inner product," $\underline{z}'_1\underline{t}/\sqrt{\underline{z}'_1\underline{z}'_1}$, therefore is the cosine of the angle between the vectors \underline{t} and \underline{z}'_1 . But

$$\underline{z}'_1\underline{t}/\sqrt{\underline{z}'_1\underline{z}'_1} = v_1/\sqrt{\underline{z}'_1\underline{z}'_1} \quad (4.3.10)$$

and hence $v_1/\sqrt{\underline{z}'_1\underline{z}'_1}$ is the cosine of the angle between the vectors \underline{t} and \underline{z}'_1 . 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{y} as "zero." We can examine each element of \underline{y} 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 i th row vector \underline{z}'_i . We shall assume that $z_{i1}, z_{i2}, \dots, z_{ip}$ are elements of \underline{z}'_i . Then \underline{t}_0 , the initial approximation to \underline{t} is obtained by normalizing \underline{z}'_i , i.e.,

$$\underline{t}_0 = \underline{z}_1' / \sqrt{\underline{z}_1' \underline{z}_1} = (t_{01}, t_{02}, \dots, t_{0p})$$

with this trial, the projections

$$\underline{v}_0 = \underline{Z} \underline{t}_0$$

are calculated and the elements v_{0i} 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:

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

$$w_i = \text{BND} \quad (\text{BND, bound, initially 8}).$$

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

$$w_i = \text{BND} - 1$$

$$\text{If } v_{0i} / \sqrt{\underline{z}_1' \underline{z}_1} > \text{BND} \times \sin \alpha/2$$

$$w_i = 0.$$

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

$$t_{1j}^* = \frac{t_{0j} - u_j}{1 - t_{0j} u_j} \quad (4.3.11)$$

$$\text{where } u_j = \frac{\sum_{i=1}^n z_{ij} v_{0i} / w_i}{\sum_{i=1}^n z_{ij}^2 / w_i} \dots \quad (4.3.12)$$

$$t_{ij} = t_{ij}^* / \sqrt{t_{i1}^* t_{i1}^*} \quad \text{are then}$$

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

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

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 $\underline{t}_1, \underline{t}_2, \dots, \underline{t}_7$. 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

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

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. $v_{\underline{t}}$ " and $(T')^{-1}\underline{t}$ is reported under the heading "Transformation vector to transfer raw data to simple structure." The corresponding elements of \underline{y} are also reported. We shall also refer to elements of \underline{y} 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

$$\mathbf{1}'Y = \mathbf{0}' \quad (4.4.1)$$

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

Also,

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

and

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

since \underline{t} is a unit vector. By virtue of the fact that $Z'Z = n_e I$, and because of (4.4.1) $(1/n_e)\underline{y}'\underline{y}$ 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{y} . There will be as many \underline{y} 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 $4 \times 3 = 12$ correlations. Now, for a given \underline{y} (a given subspace),

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

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

$$(1/n_e)\mathbf{Y}'\mathbf{y} = \mathbf{T}\mathbf{t} \quad (4.4.5)$$

Hence the estimated covariances between the original variables and a single artificial variable, are elements of $\mathbf{T}\mathbf{t}$. 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

$$(1/n_e)\mathbf{y}'\mathbf{y} = 1 \quad (4.4.6)$$

since a sum of squares ($\mathbf{y}'\mathbf{y}$) must be divided by degrees of freedom to produce a variance estimate. Hence to obtain the correlations, we have to divide the elements of $\mathbf{T}\mathbf{t}$ by the square root of the estimate of the variance of the observable variable only. In the computer program, after the vector \mathbf{t} is obtained, the product $\mathbf{T}\mathbf{t}$ is formed and the correlations are then calculated by division of each of the elements of $\mathbf{T}\mathbf{t}$ 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. \mathbf{T} is also stored.

Once the correlations between the original variables and their scores with reference to a subspace (i.e., vector \underline{y}) 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{y} plays a similar role in the sense that the elements of a well defined subspace represented by \underline{y} has most elements near zero (see section 4.3) and a few far removed from zero. In analogy with discriminant analysis, \underline{y} 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, CLUSTER, 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 2001. 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 CLUSTER program, he should next run the IEBGENER program. This program supplies the output of CLUSTER 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 CLUSTER program, is put before the punched deck produced by the CLUSTER 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_c , 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 subroutine 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

CORIS 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

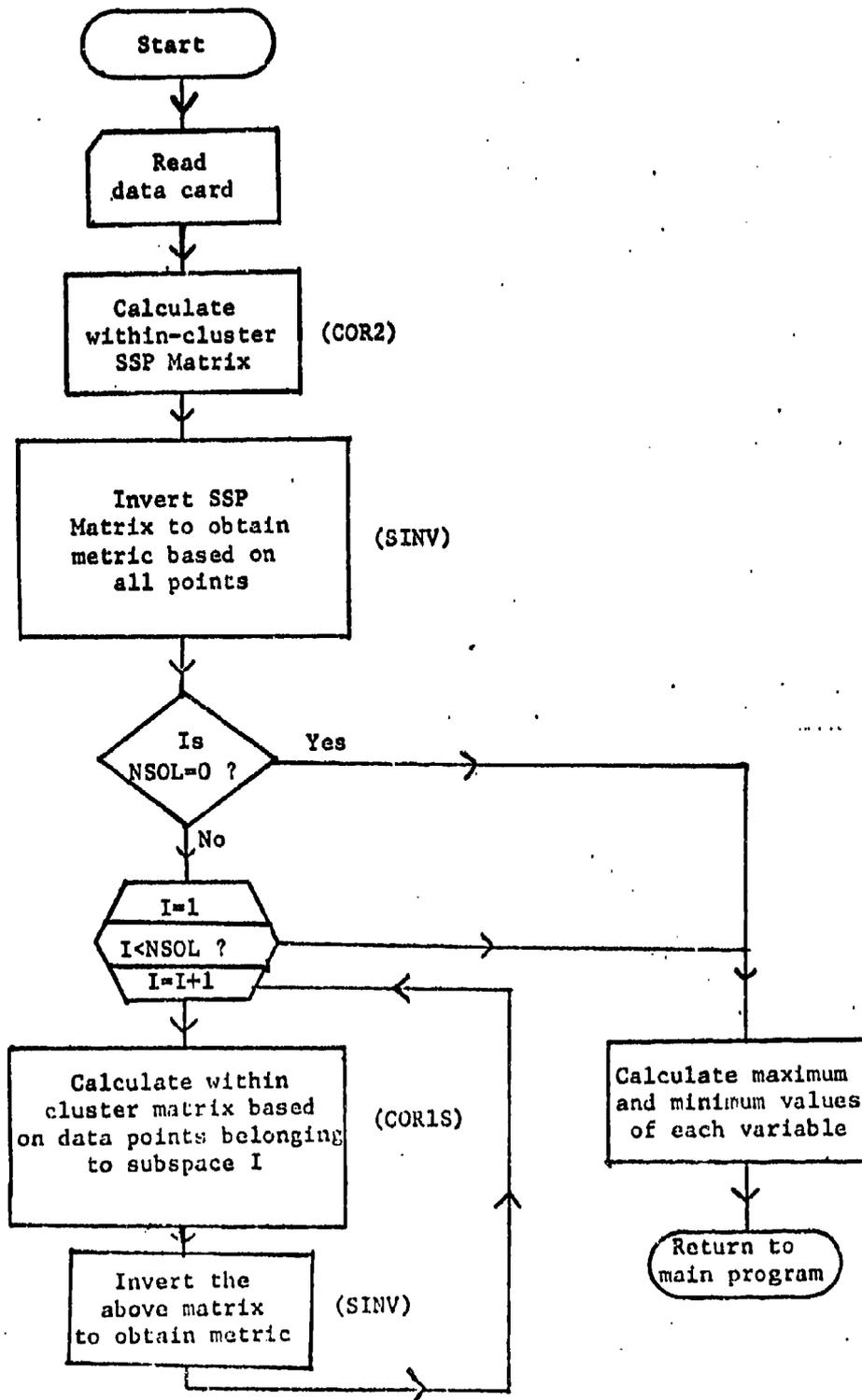
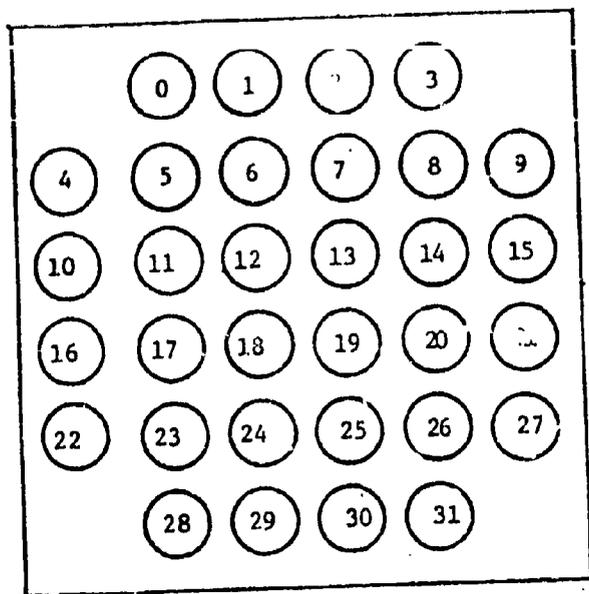
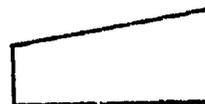
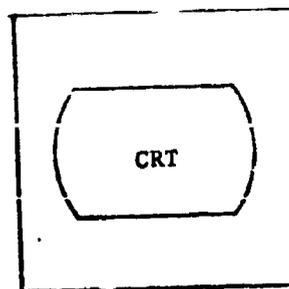


Figure 5.4.1

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 NCG (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



Alphanumeric 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

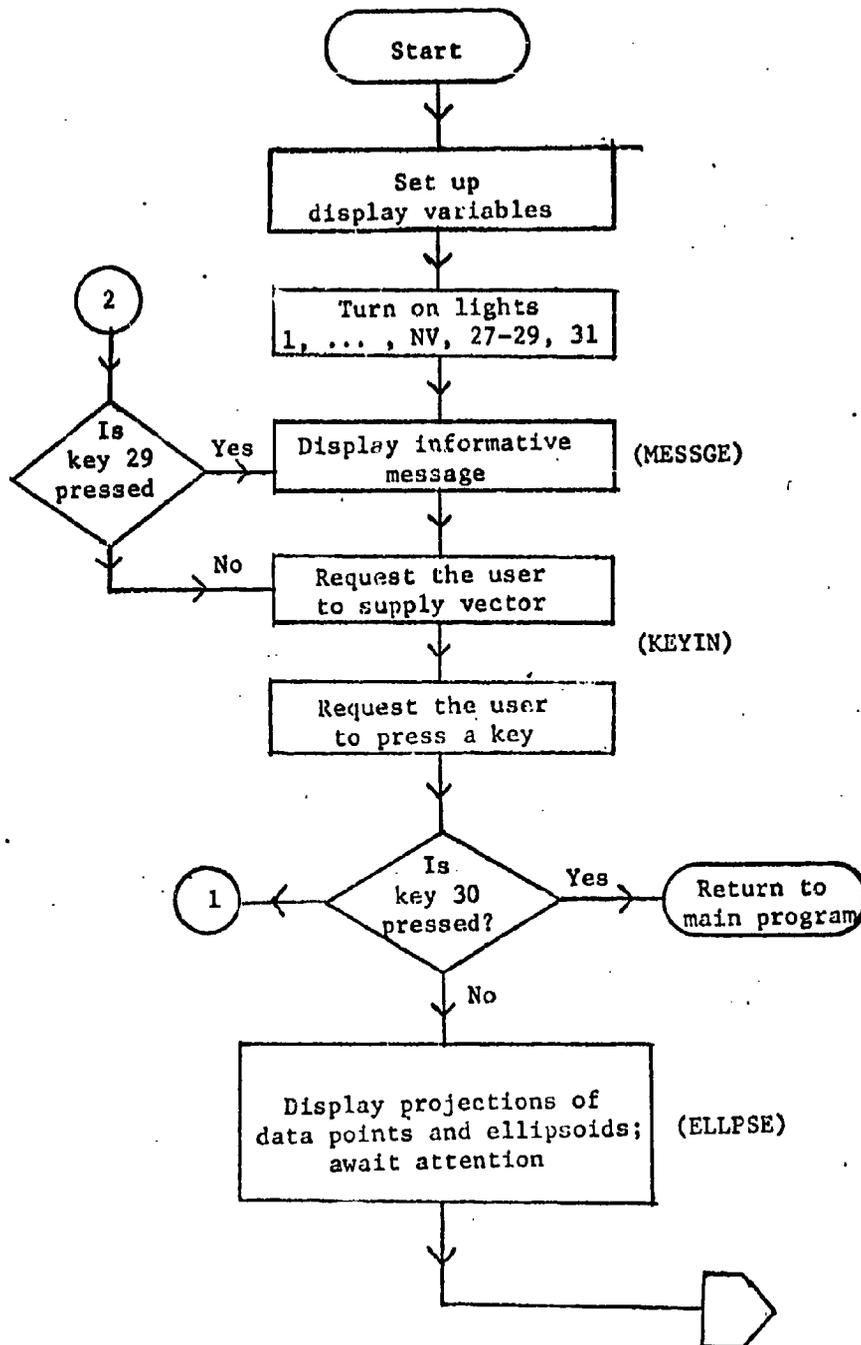


Figure 5.4.2a

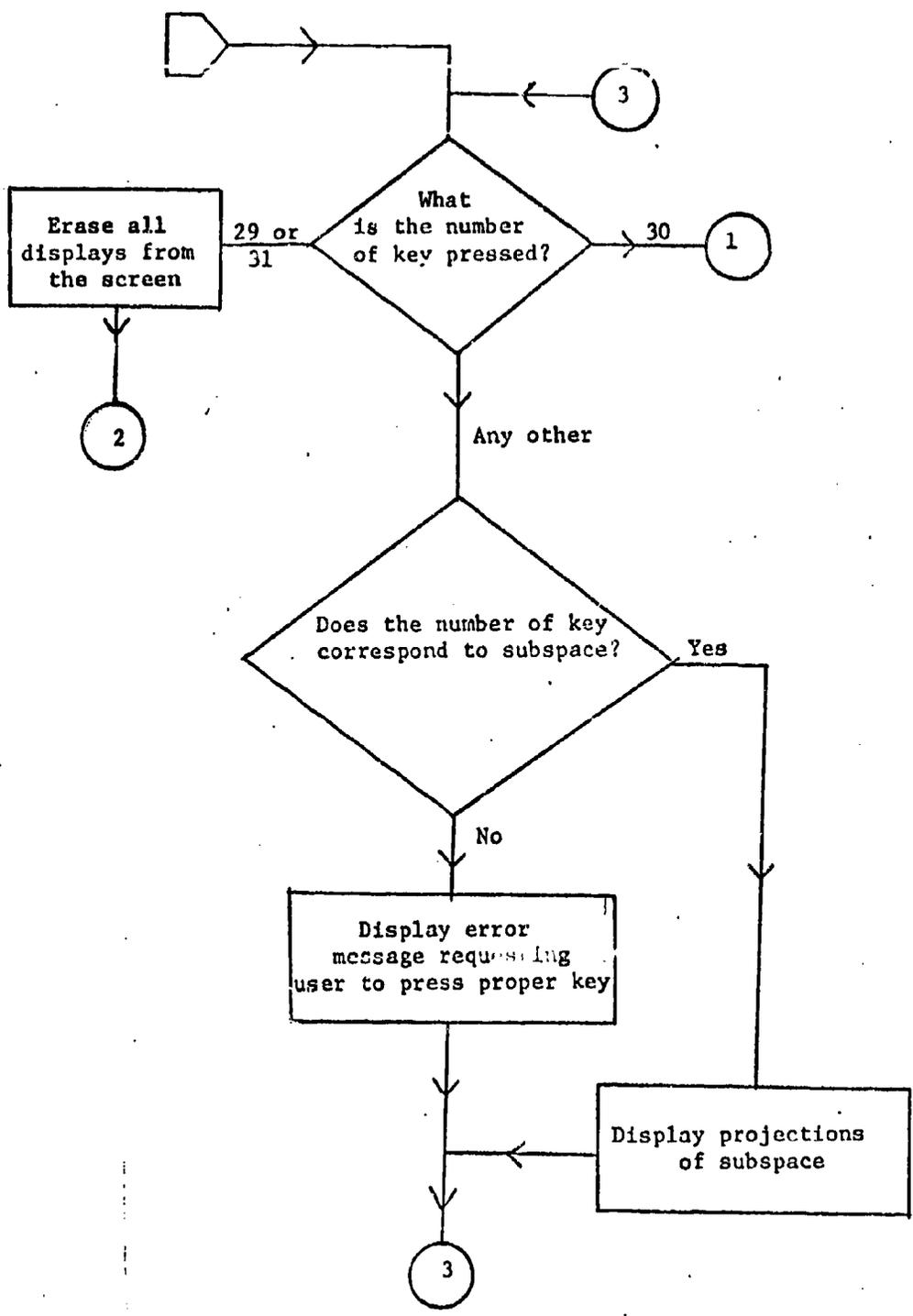


Figure 5.4.2b

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 (CLUSTER) 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

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

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

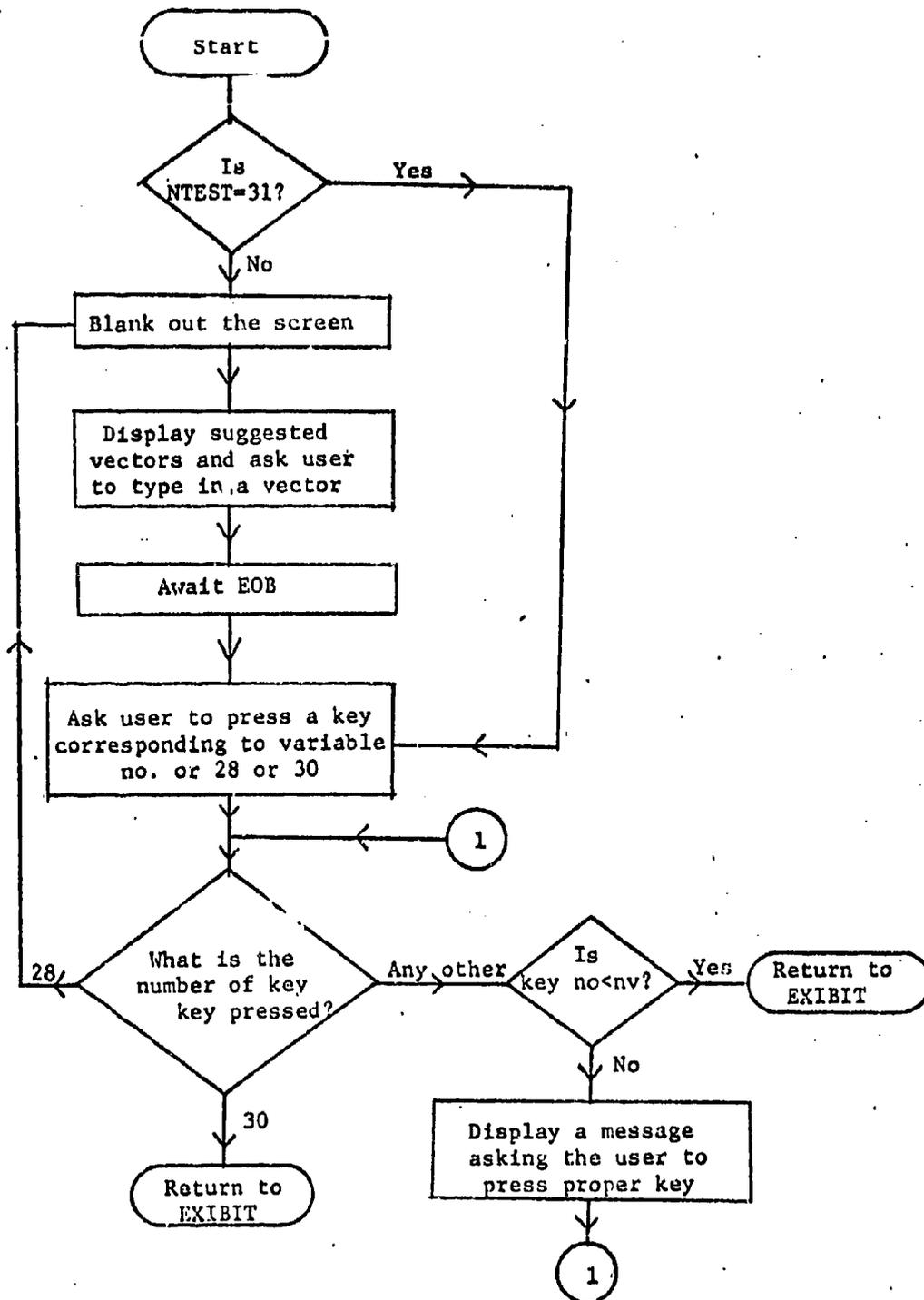


Figure 5.4.3

with the desired variable. If the user supplies a vector collinear with his 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 $\underline{\mu}'_1 R$. The points describing the circumference of the ellipses

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

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 i th subspace is denoted by S_i^{-1} , the subroutine calculates the points describing the circumference of ellipses

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

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.

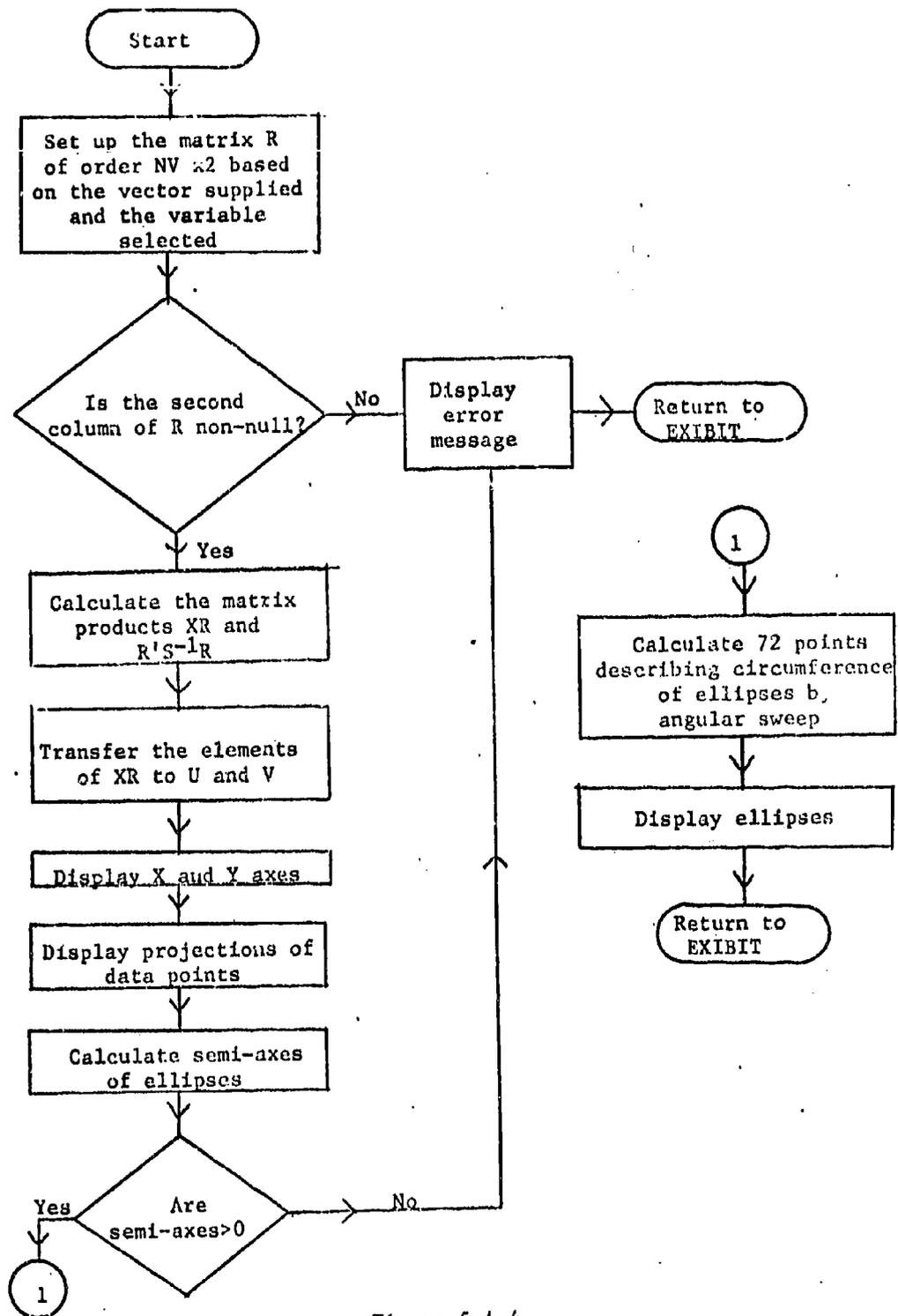
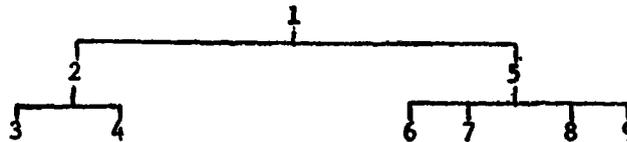


Figure 5.4.4

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, SCNDVTK 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 DSN=NAME=&&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

//FORT.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

/*

//STEP0 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.GRAFLIB(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(RFSET)
OVERLAY TWA
INCLUDE MODULE (LINKAS)
OVERLAY TWA
```

INCLUDE MODULE (LINKA9)

INCLUDE SYSLIB(SCNDVDK)

INCLUDE SYSLIB(CUR\$\$)

INCLUDE SYSLIB(BUFRS)

INCLUDE SYSLIB(SCTDV)

INCLUDE SYSLIB(RCUR\$)

INCLUDE SYSLIB(READSC)

OVERLAY TWA

INCLUDE MODULE (LINKA6)

OVERLAY TWA

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 CLUSTER 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) //STEP 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 CLUSTER 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 CLUSTER 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, number 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 IEDGENER?

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

-0.309 -0.275 0.256 -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) =

IF 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 alphanumeric 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 alphanumeric 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 $(1.0, 0.0, \dots, 0.0)$ 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 $(a, 0, 0, \dots, 0)$ 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 45×4 matrix, numbered column-wise. The first 25 measurements on the 4th

variable were then redefined by the relation

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

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

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

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

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

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)

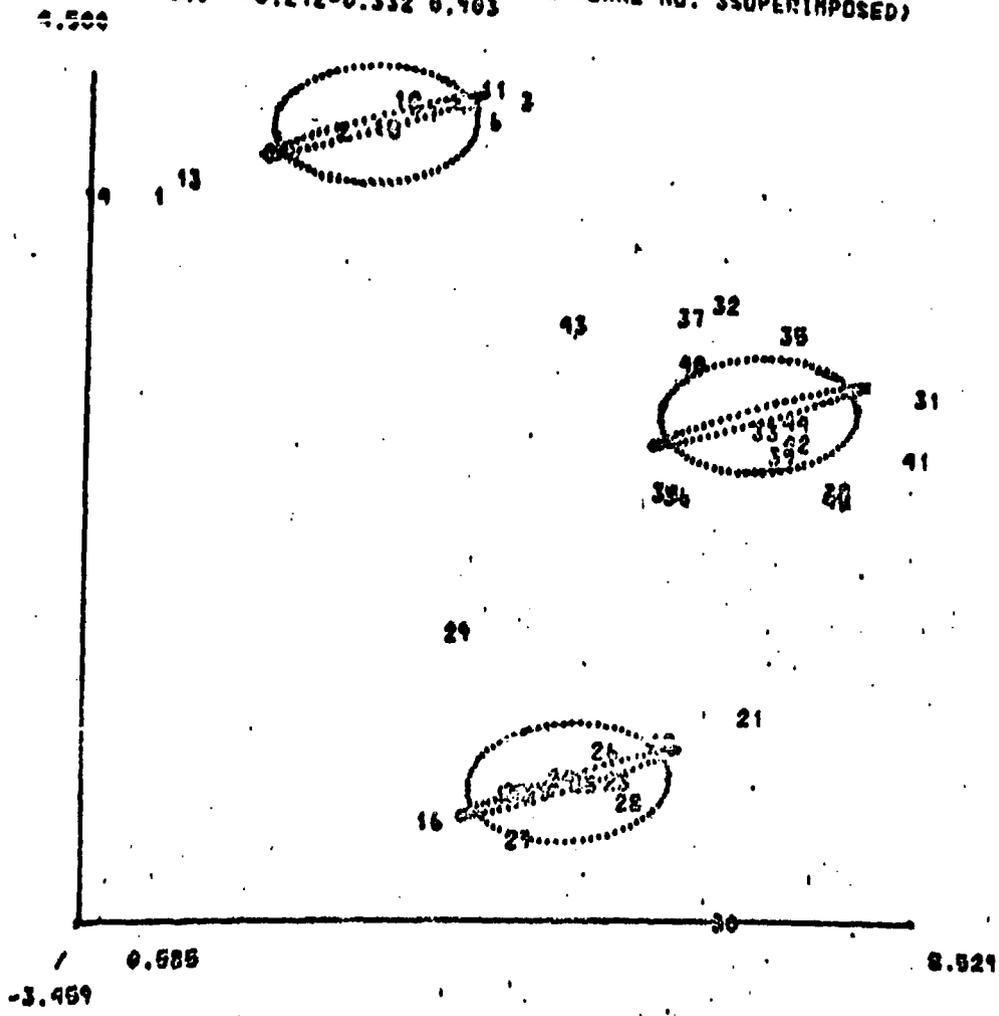
Plane 3: --1, 2, 4, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, (20 points)

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 CLUSTER 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.

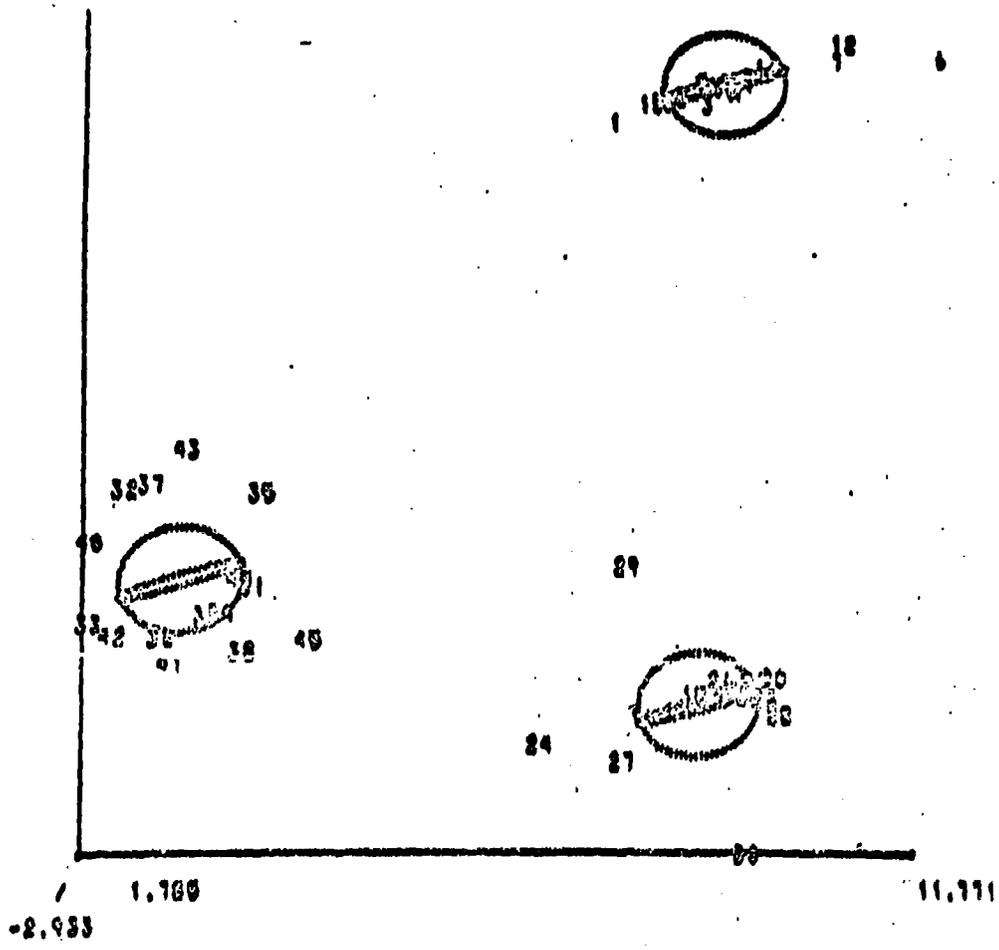
VARIABLE 1 AGAINST VECTOR BELOW (SIMPLE PLANE NO. 3 SUPERIMPOSED)
0.0 -0.272 -0.332 0.403



(b) Variable 2 against vector 3--After normalizing, the vector reduced to (0.317, 0.0, 0.327, 0.690). 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.

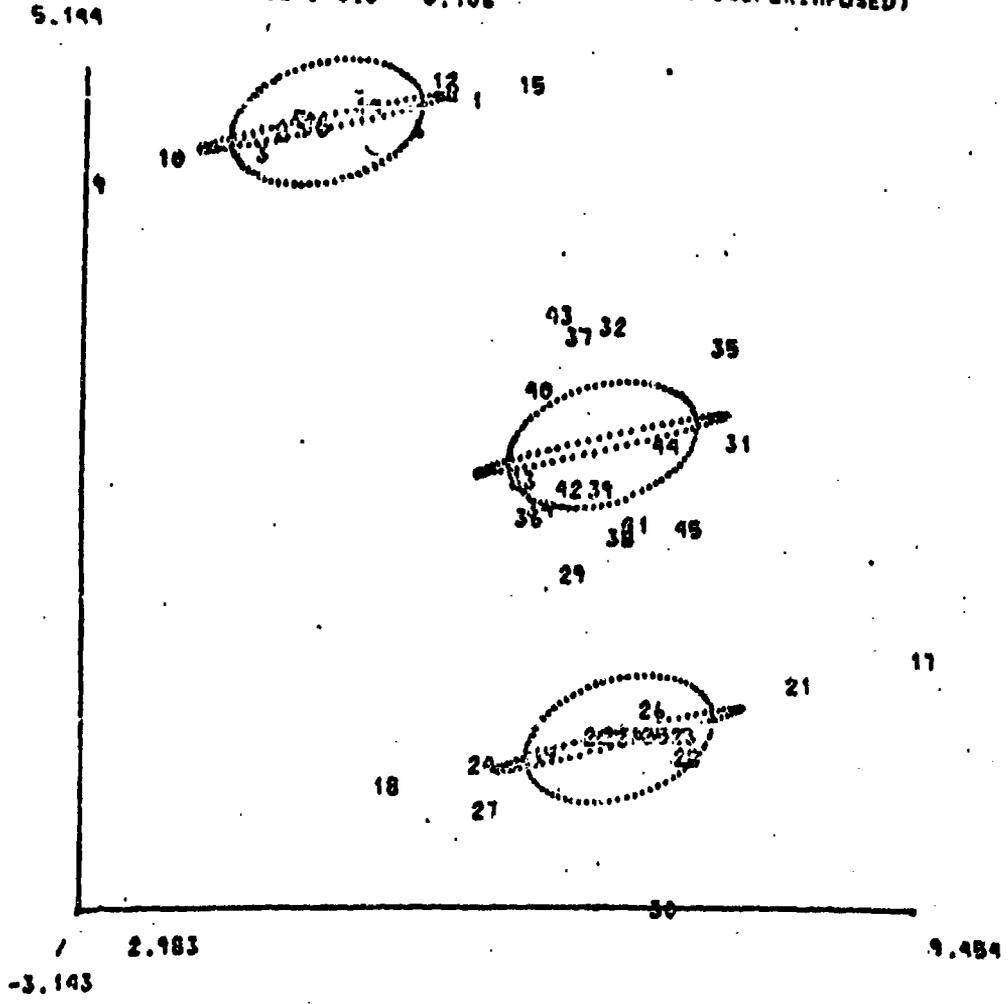
VARIABLE AGAINST VECTOR DELTA (SIMPLE PLANE NO. SUPERIMPOSED)

6.058



(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.

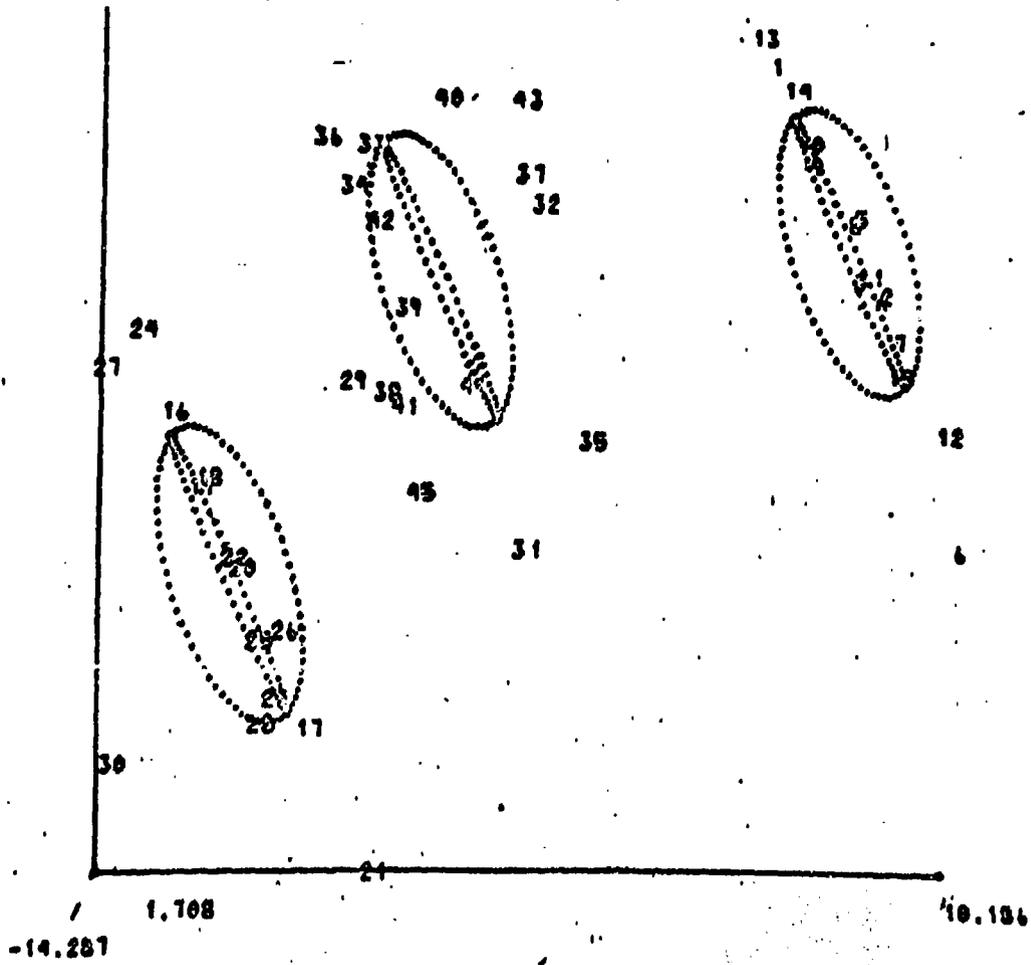
VARIABLE AGAINST VECTOR DELTA, (SIMPLE PLANE NO. 3 SUPERIMPOSED)
-0.522 -0.213 0.0 0.906



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

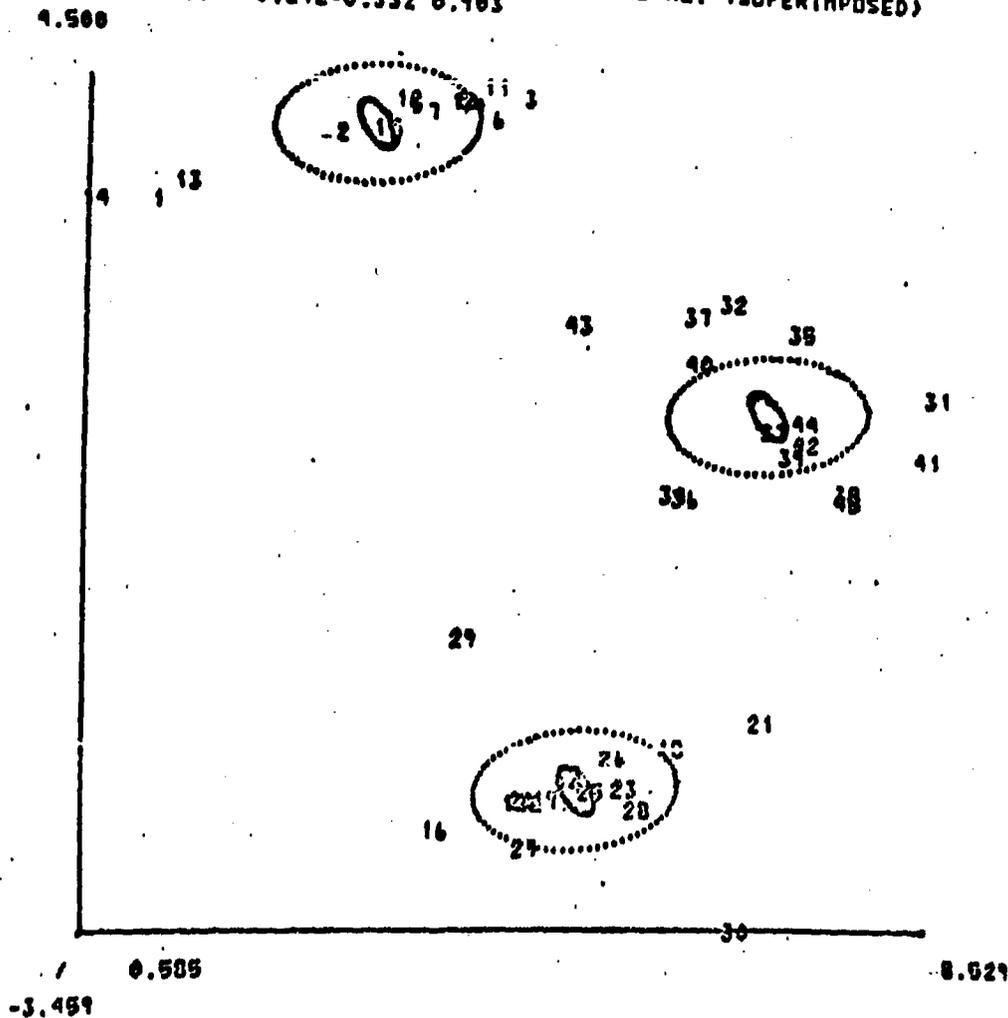
-8.062



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.

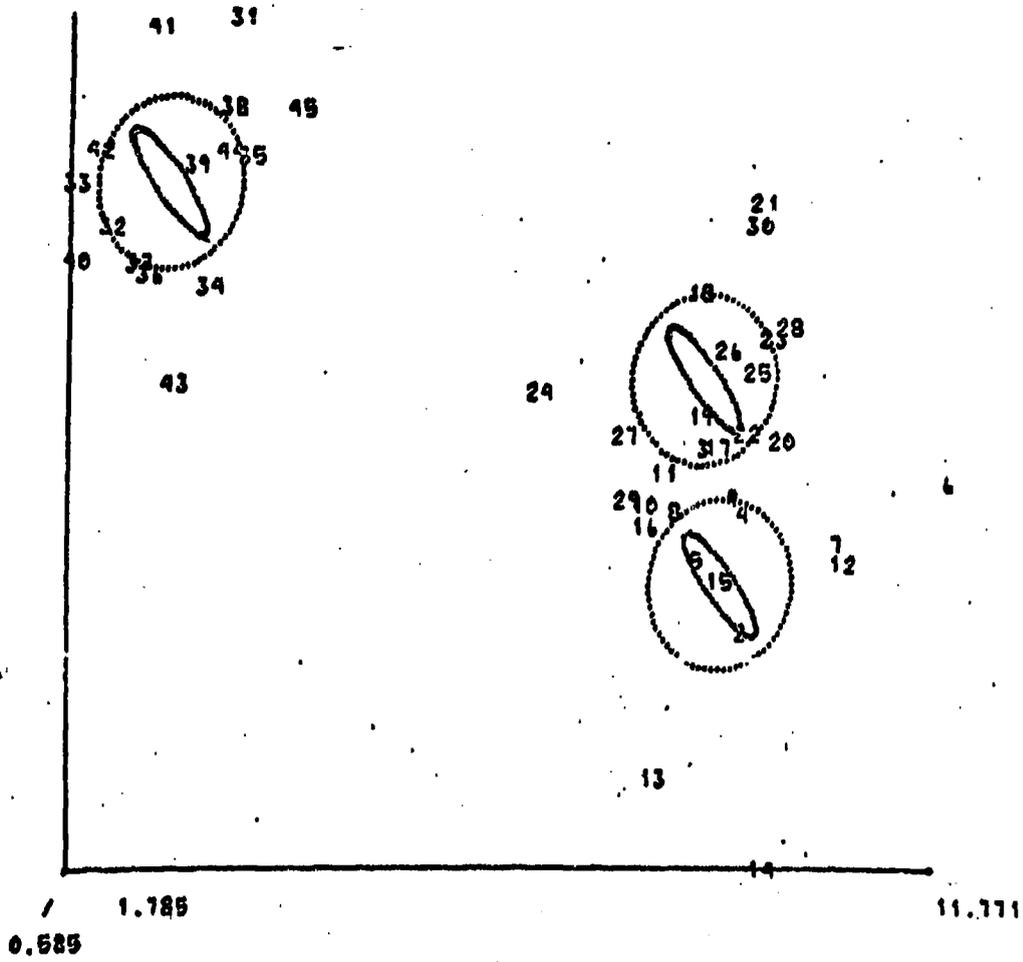
VARIABLE 1 AGAINST VECTOR BELOW (SIMPLE PLANE NO. 1 SUPERIMPOSED)
0.0 -0.212 -0.332 0.103



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

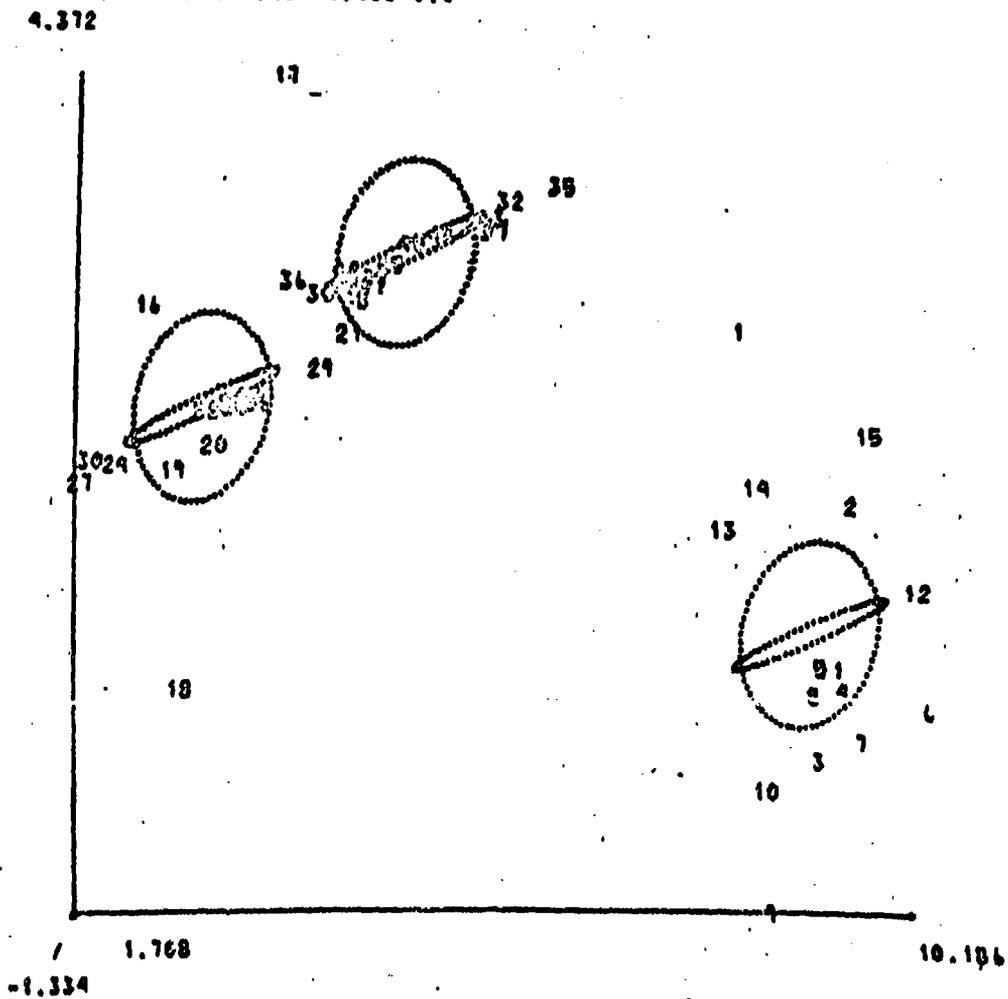
8.529



(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 RELON. (SIMPLE PLANE NO. 1 SUPERIMPOSED)

-0.325 -0.289 0.400 0.0

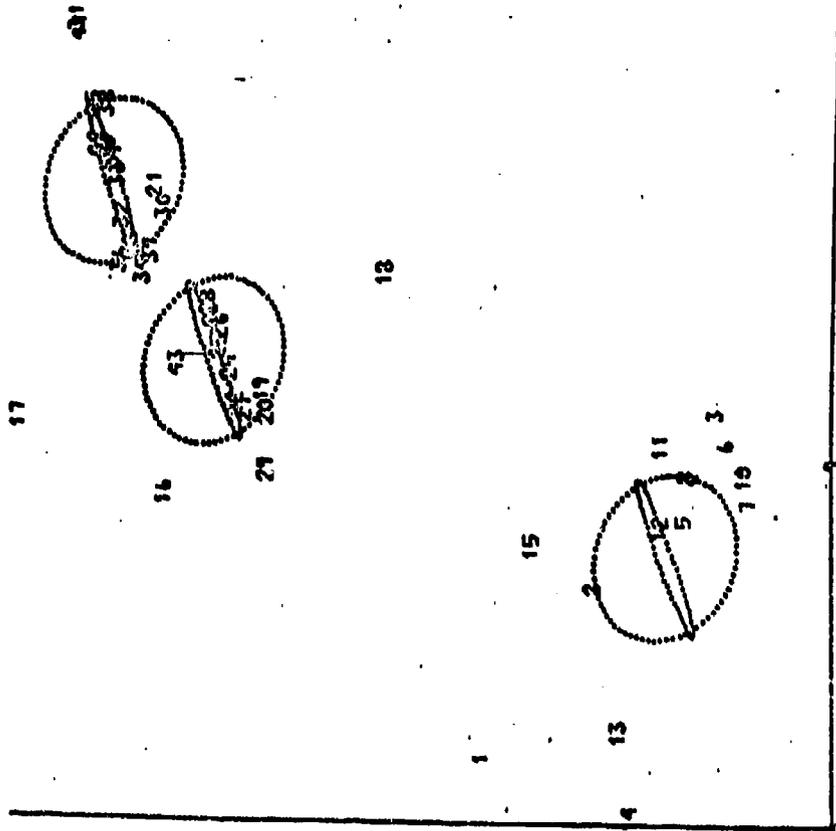


The user should find the other displays easy to understand. A good simple structure is seen only when a plane is selected which passes through a vector that transforms the raw data into a simple structure solution, and the corresponding simple structure plane is superimposed. It should, however, be noted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

VARIABLE AGAINST VECTOR BELOW: (SIMPLE PLANE NO. 1 SUPERIMPOSED)

0.6 -0.281 0.100-0.325

4.632



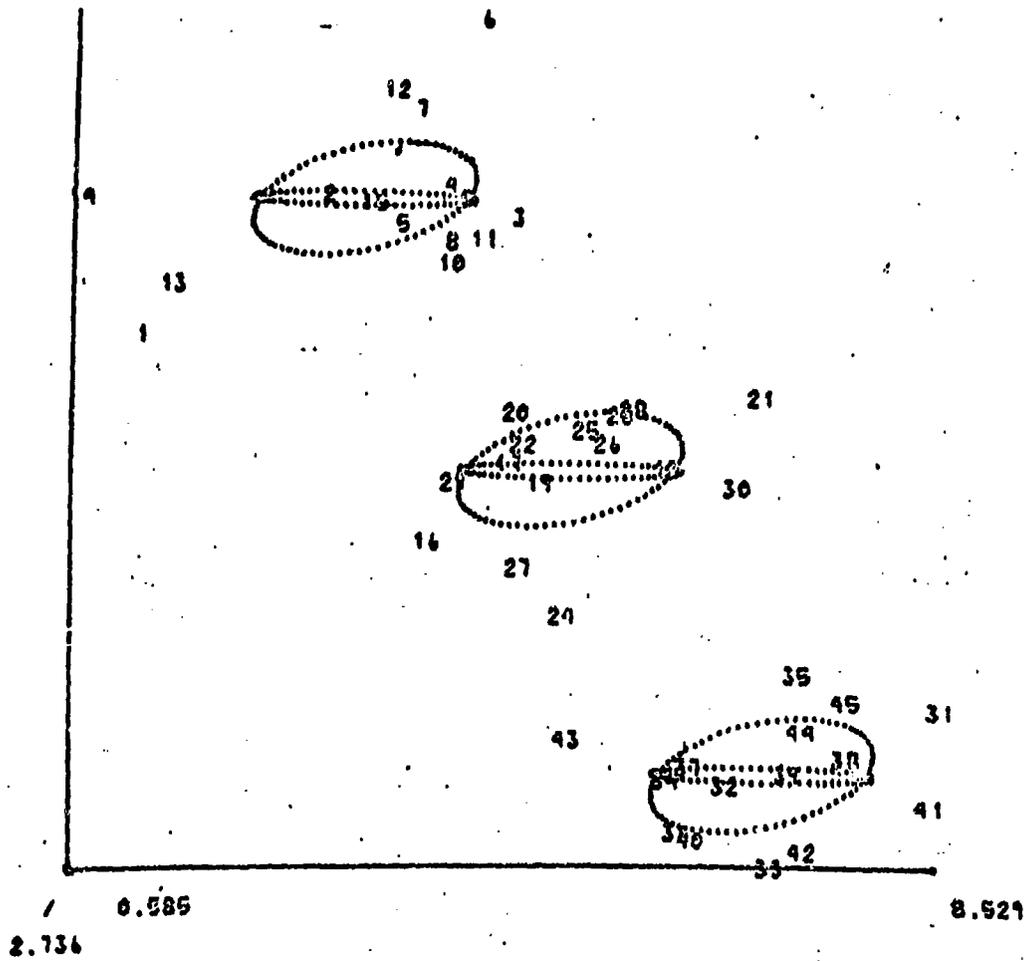
0.585

-2.821

8.524

VARIABLE AGAINST VECTOR BELOW, (SIMPLE PLANE NO. 2 SUPERIMPOSED)
0.0 0.843-0.155 0.516

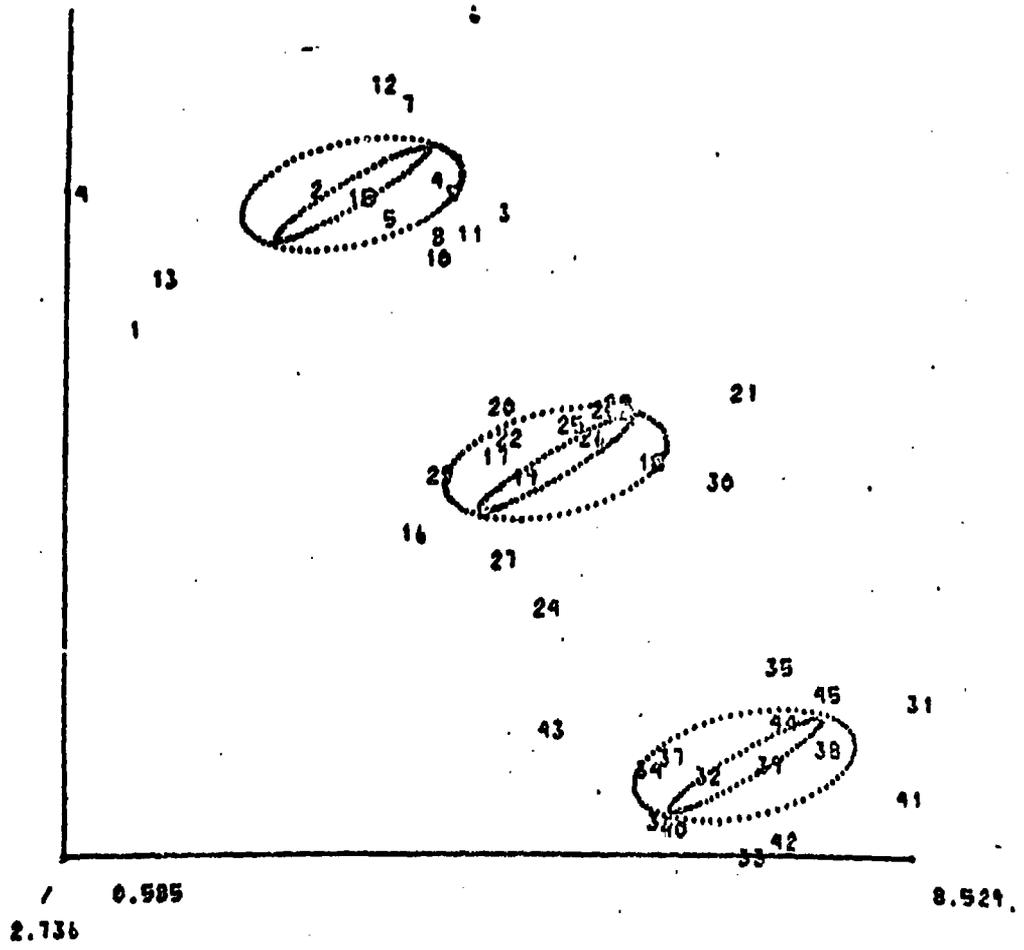
14.334



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

0.0 0.843-0.155 0.516

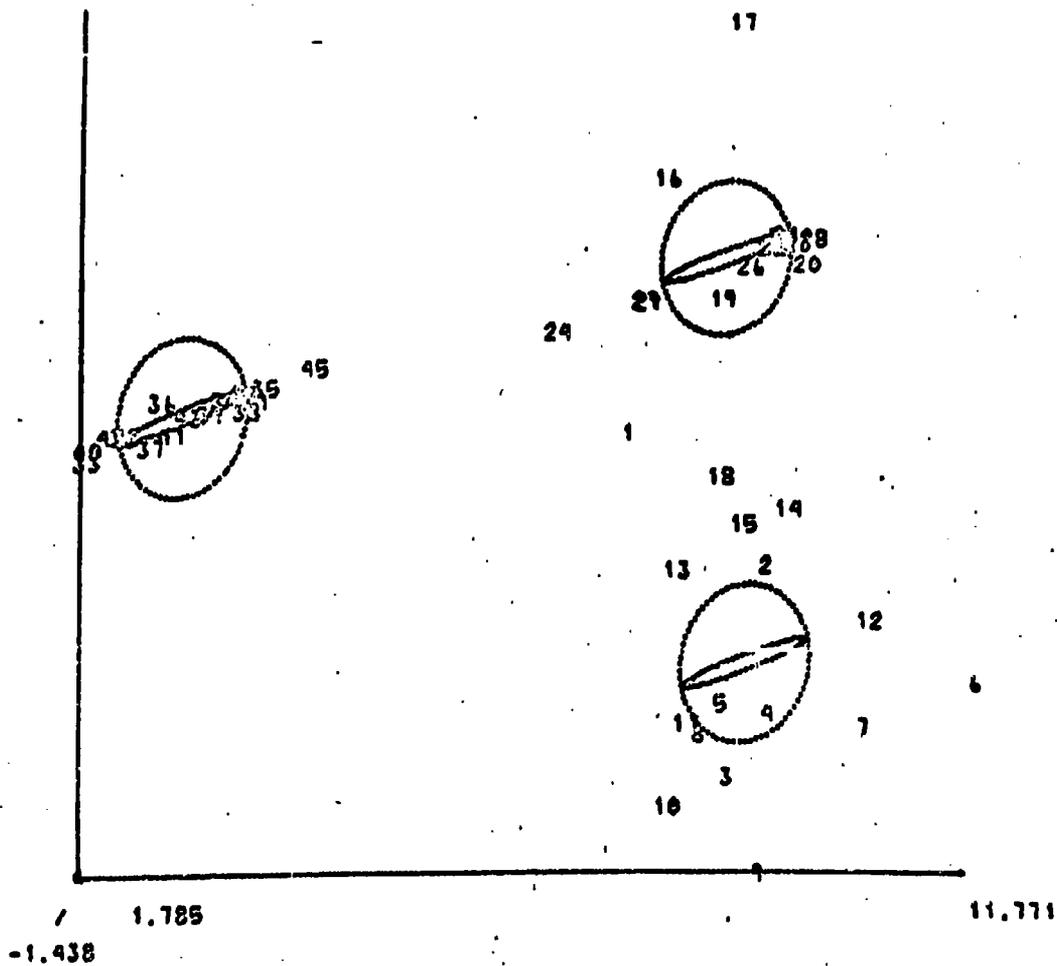
14.334



VARIABLE 2 AGAINST VECTOR BELOW. (SIMPLE PLANE NO. 1 SUPERIMPOSED)

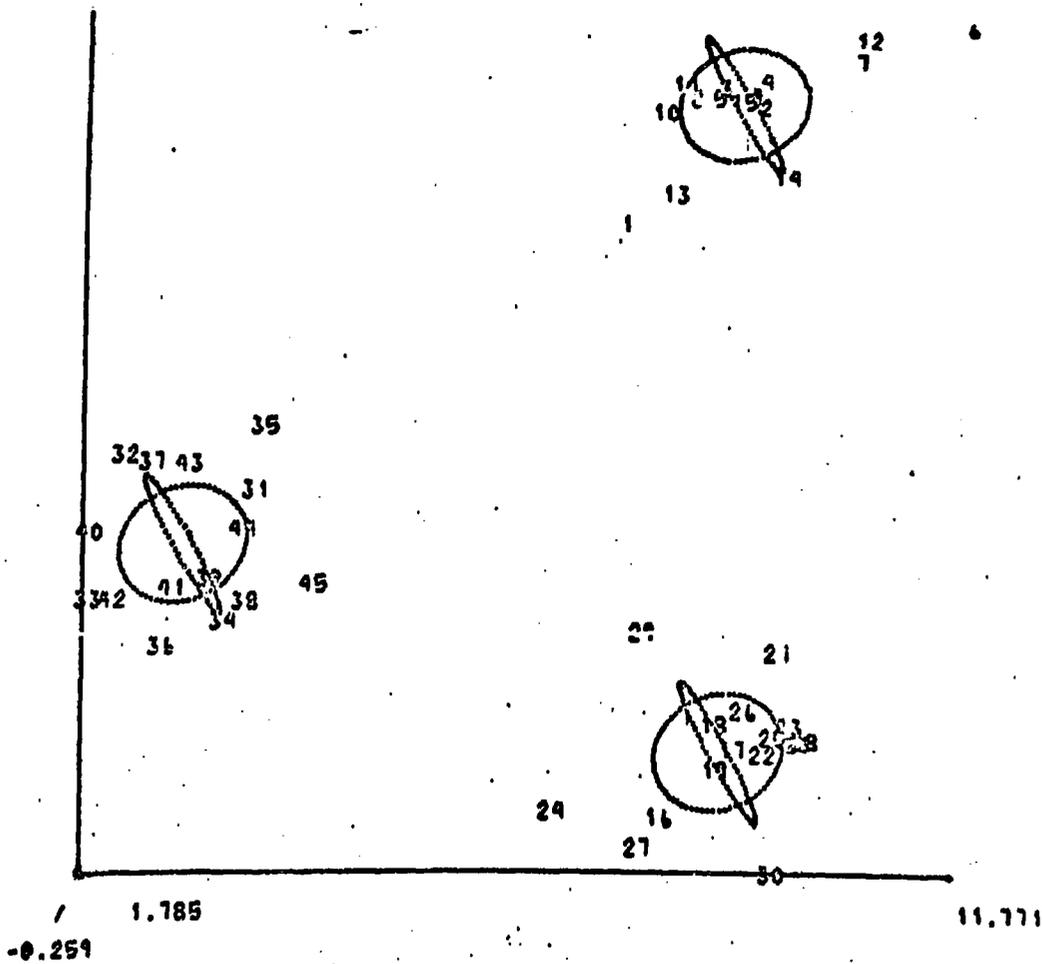
-0.322 0.0 0.891-0.322

5.751



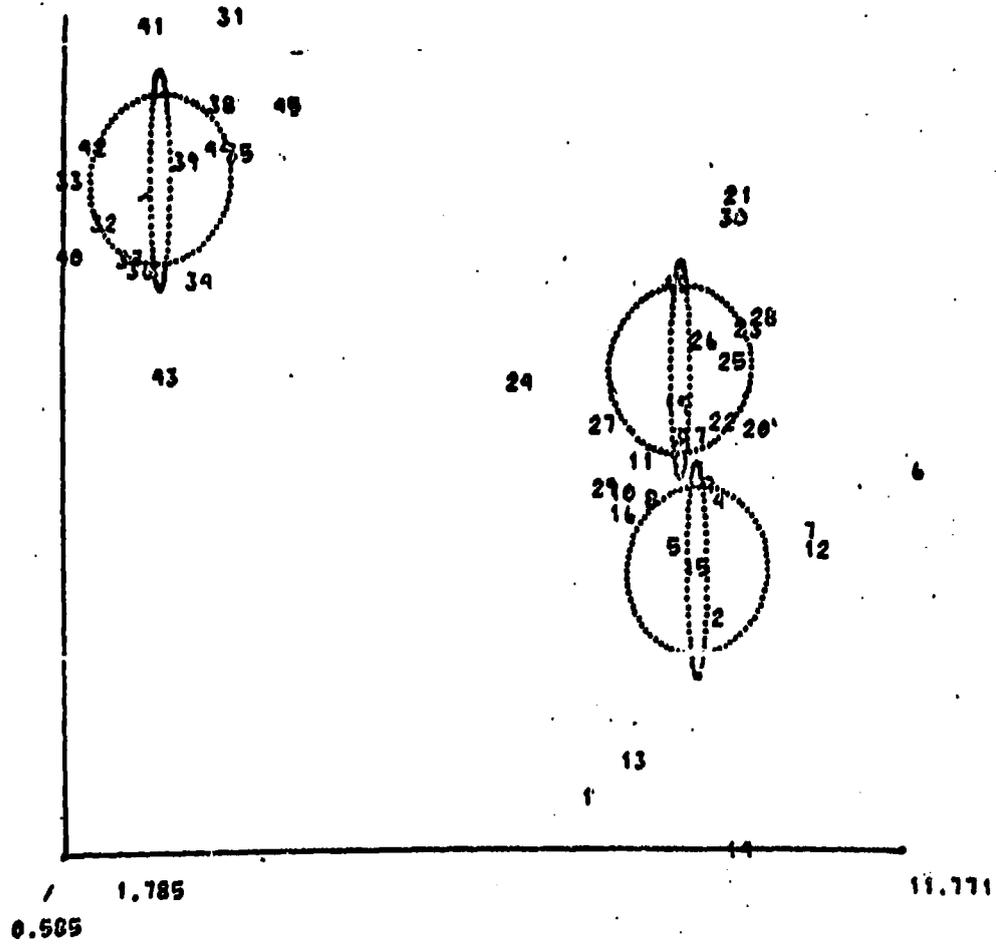
VARIABLE 2 AGAINST VECTOR BELOW (SIMPLE PLANE NO. 2 SUPERIMPOSED)
0.022 0.0 -0.208 0.957

8.293



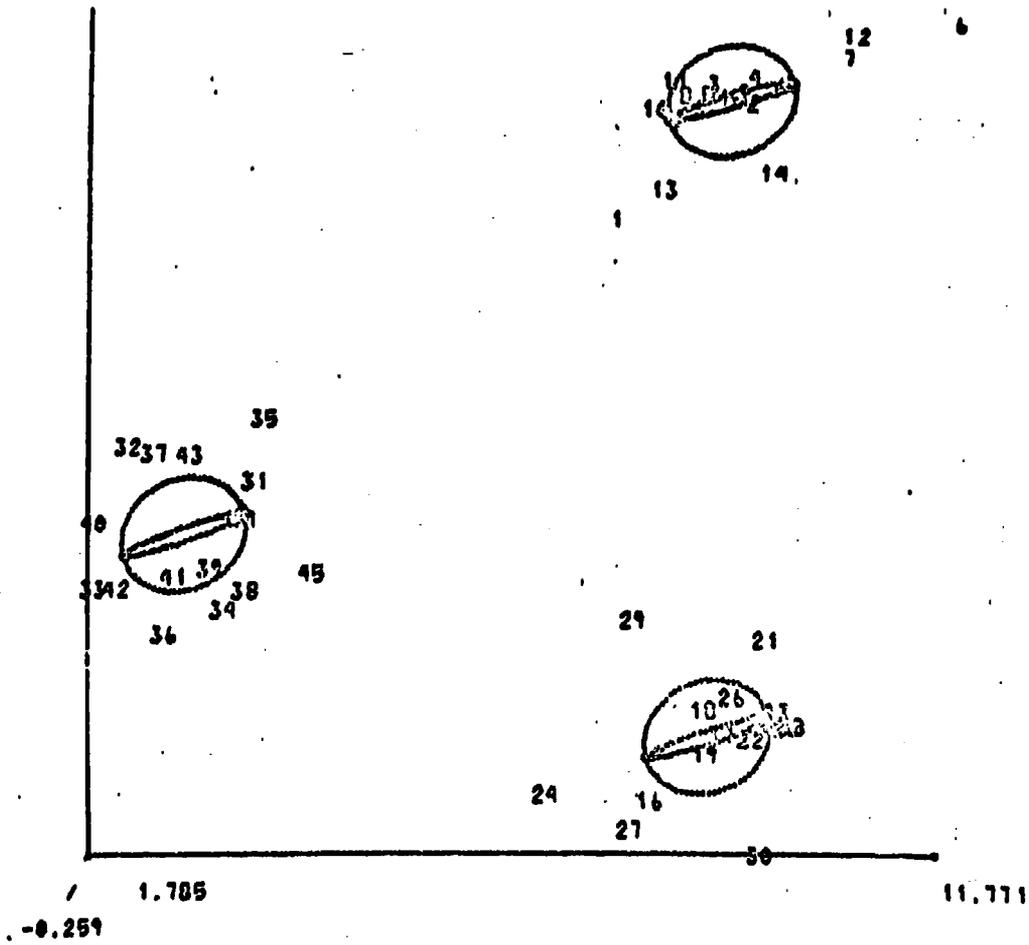
VARIABLE AGAINST VECTOR BELOW. (SIMPLE PLANE NO. 2 SUPERIMPOSED)
1.000 0.0 0.0 0.0

8.529

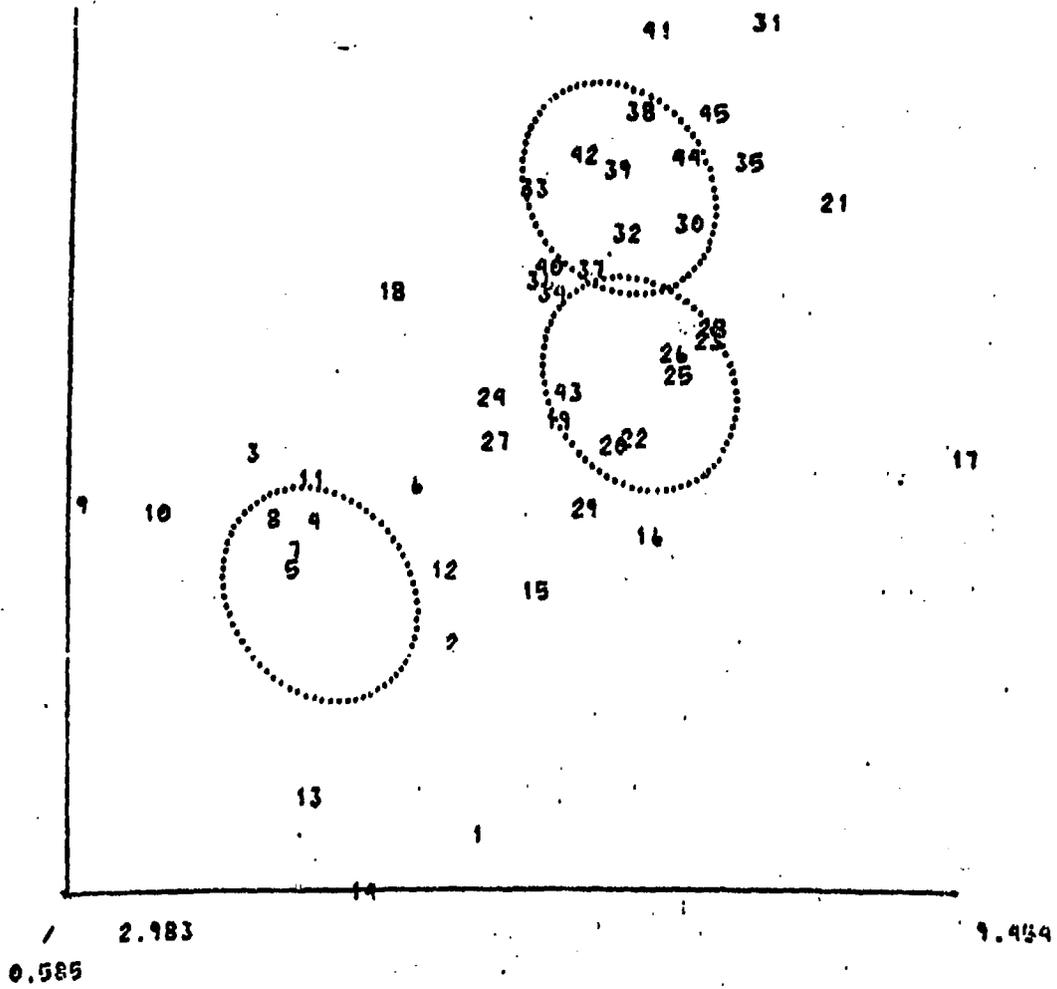


VARIABLE AGAINST VECTOR BELOM. (SIMPLE PLANE NO. SUPERIMPOSED)
0.022 0.0 -0.250 0.151

8.293

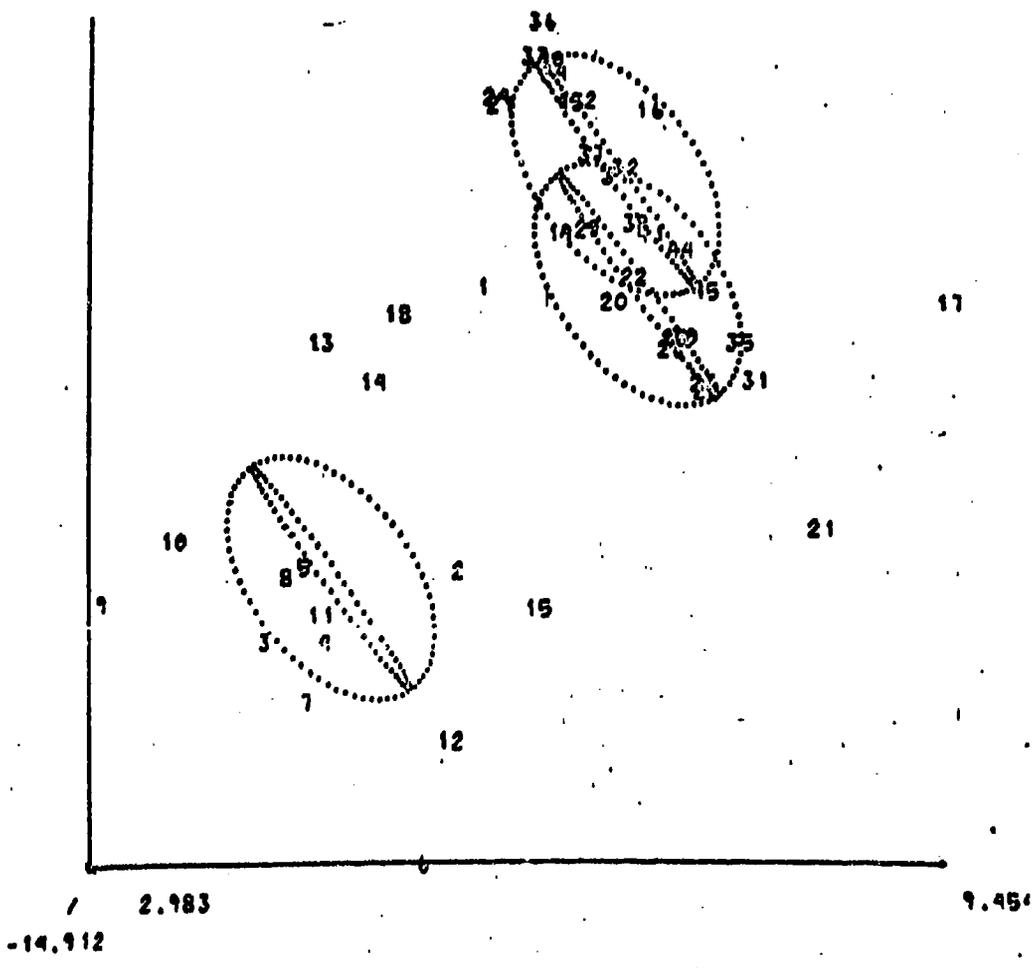


VARIABLE AGAINST VECTOR
1.000 0.0 0.0 0.0
8.529



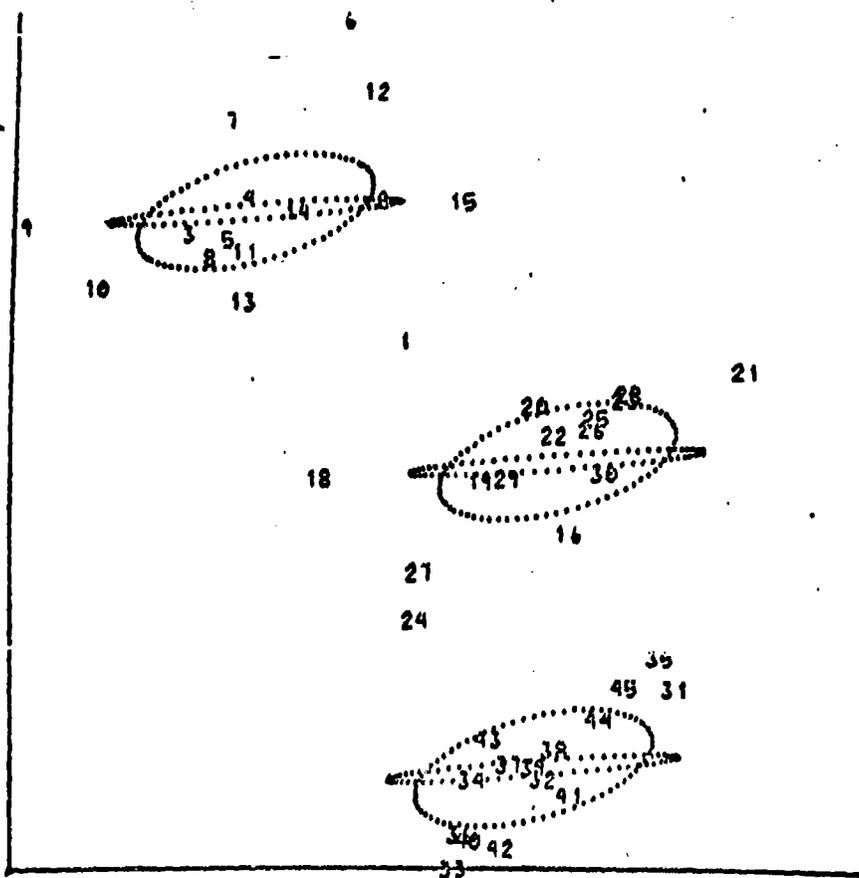
VARIABLE AGAINST VECTOR BELDW. (SIMPLE PLANE NO. 1 SUPERIMPOSED)
-0.598 -0.533 0.0 -0.598

-7.359



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

15.408

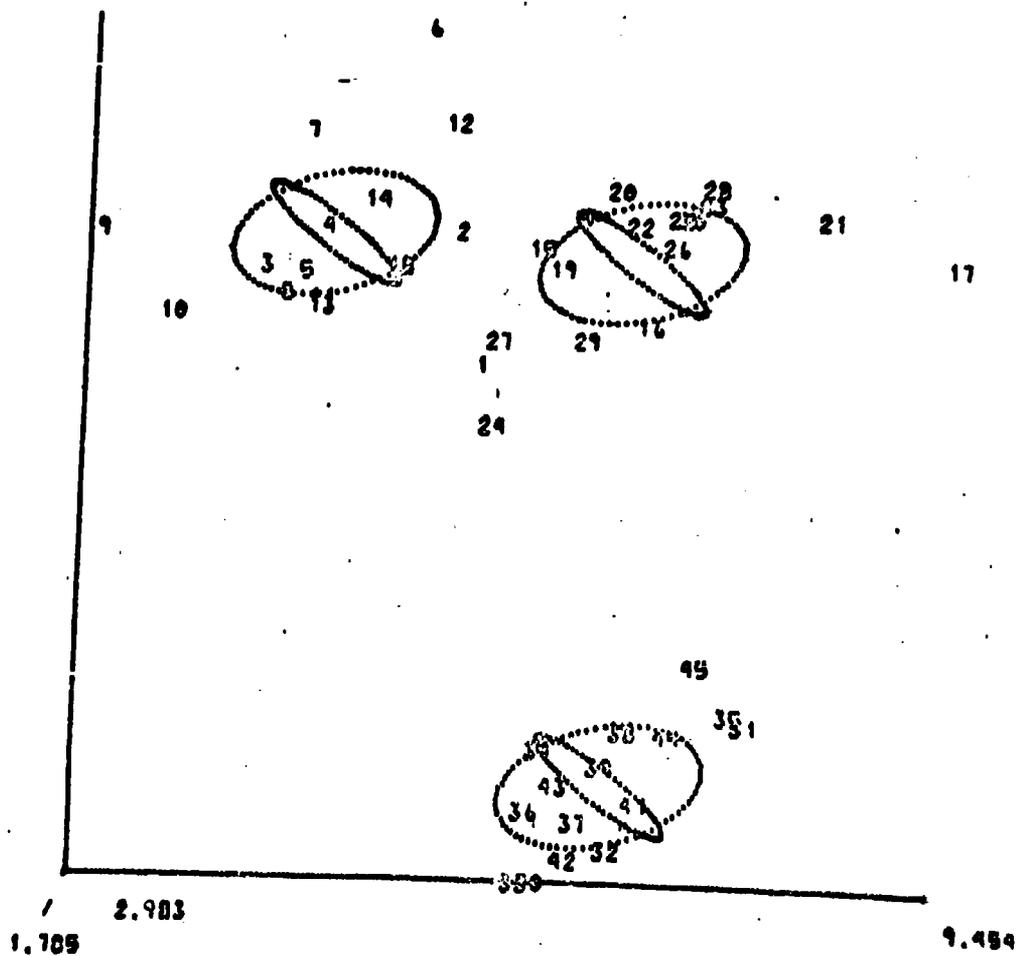


2.983
3.841

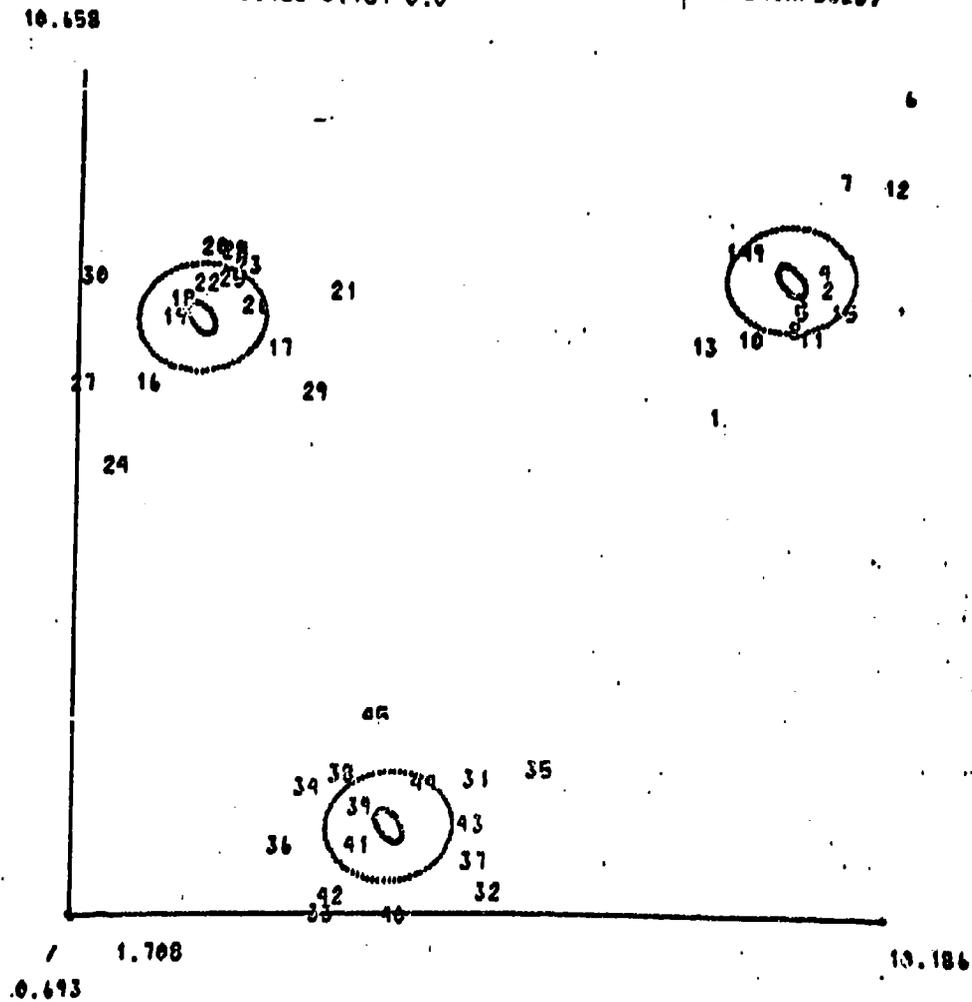
9.454

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

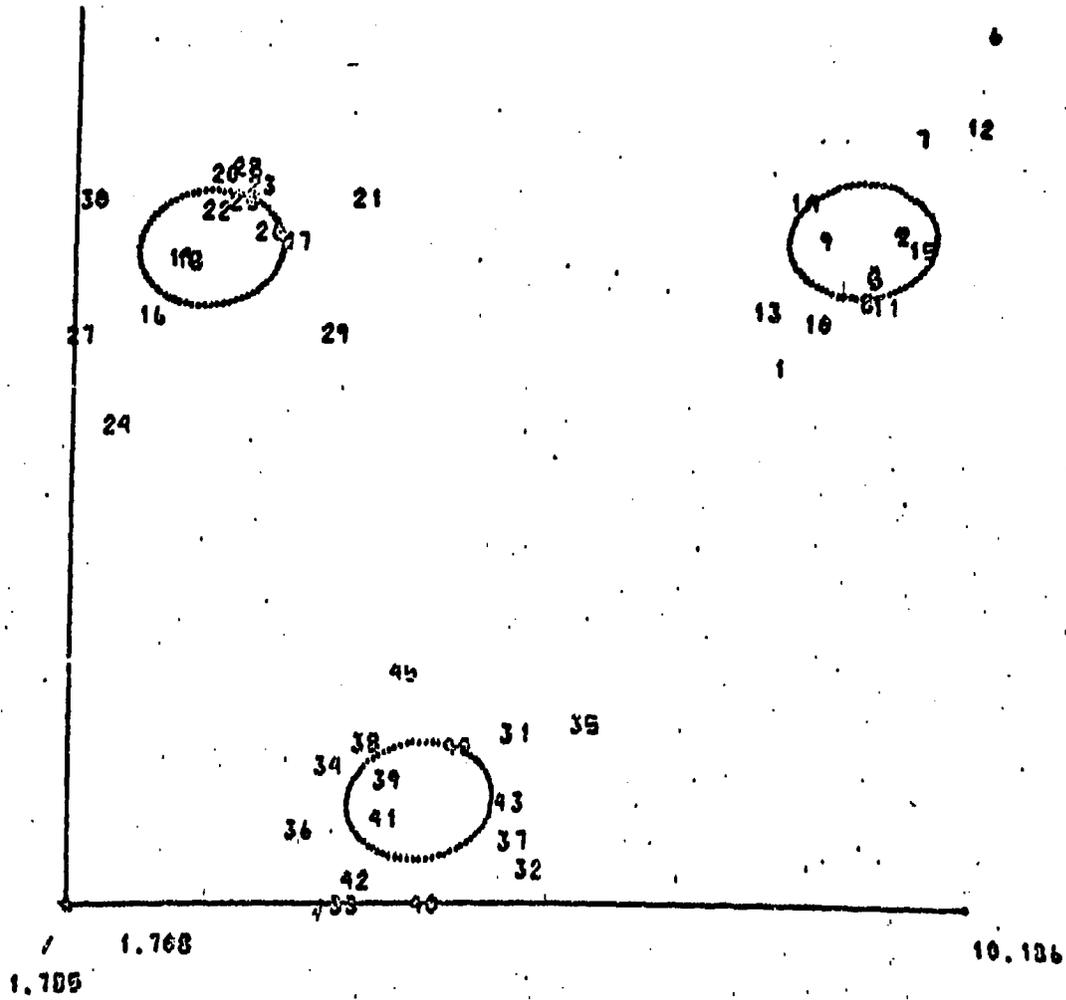
11.771



VARIABLE 4 AGAINST VECTOR BELOW (SIMPLE PLANE NO. 2 SUPERIMPOSED)
0.019 0.183-0.181 0.0



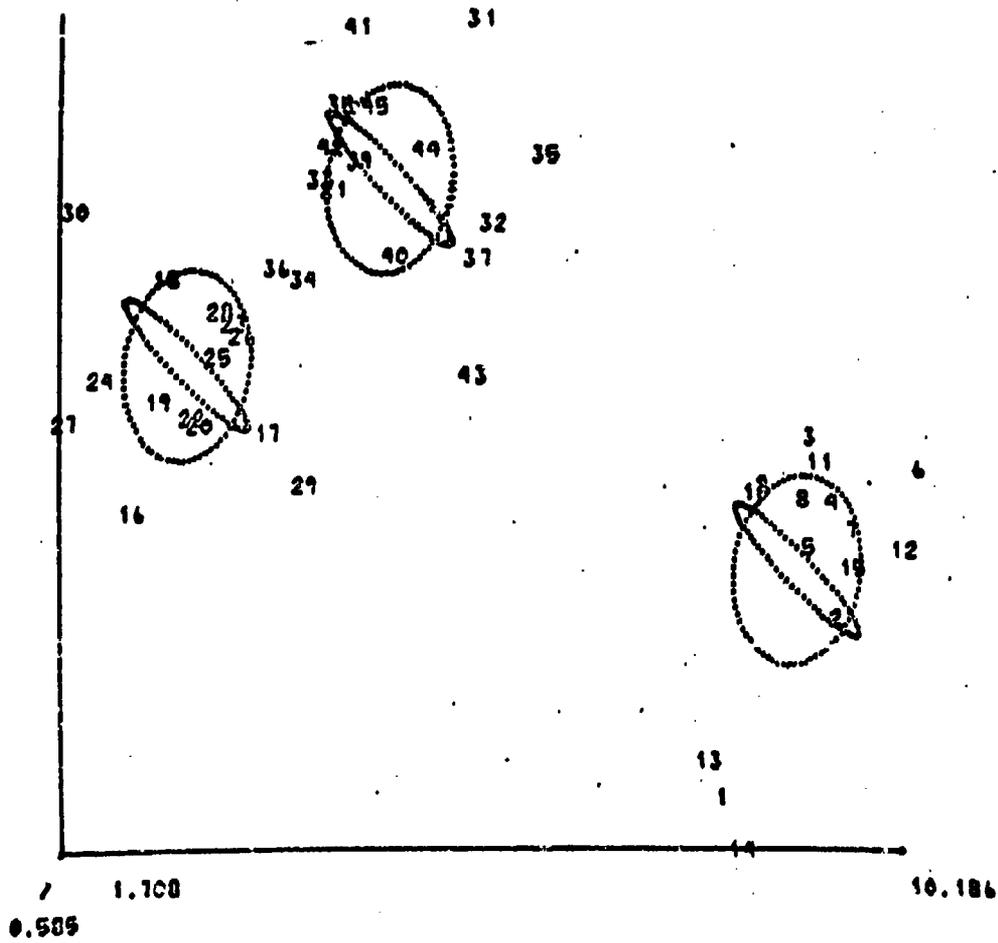
VARIABLE AGAINST VECTOR
0.0 1.000 0.0 0.0
11.771



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

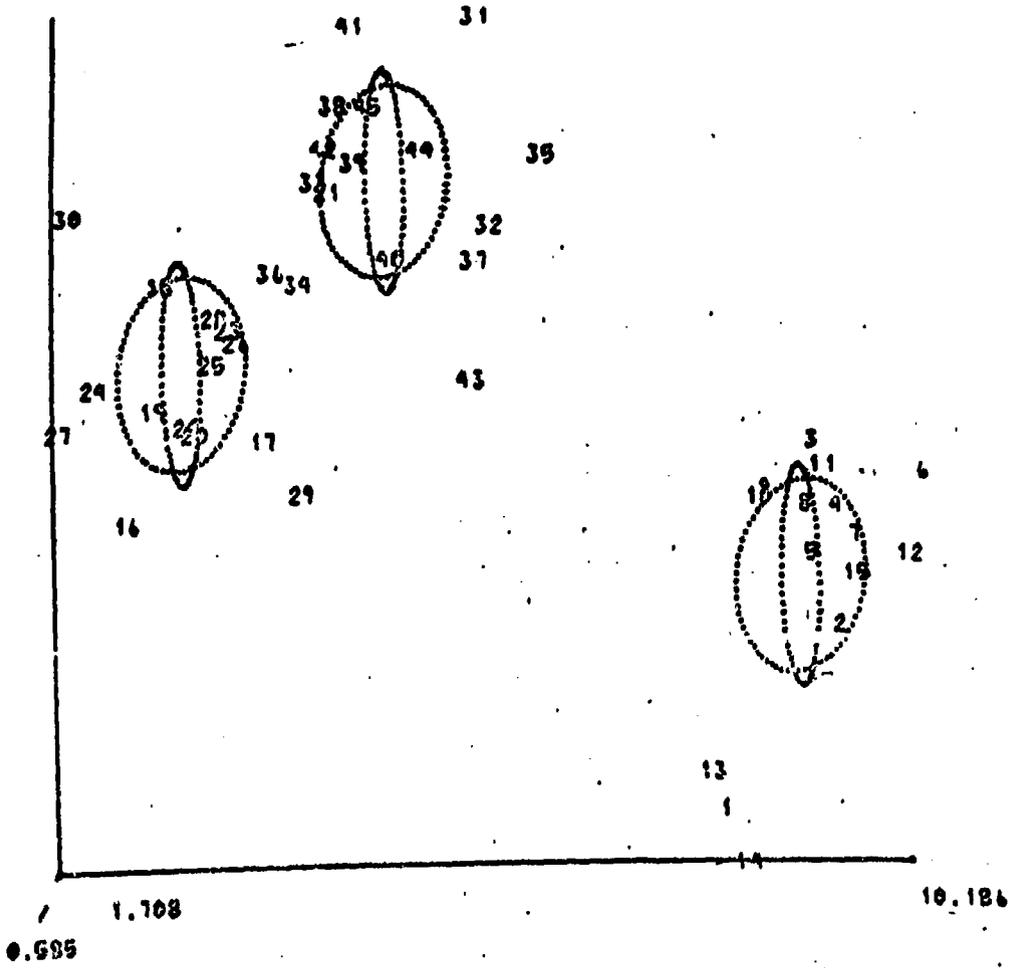
1.000 0.0 0.0 0.0

8.529



VARIABLE 4 AGAINST VECTOR BELOW, (SIMPLE PLANE NO. 2 SUPERIMPOSED)
1.080 0.0 0.0 0.0

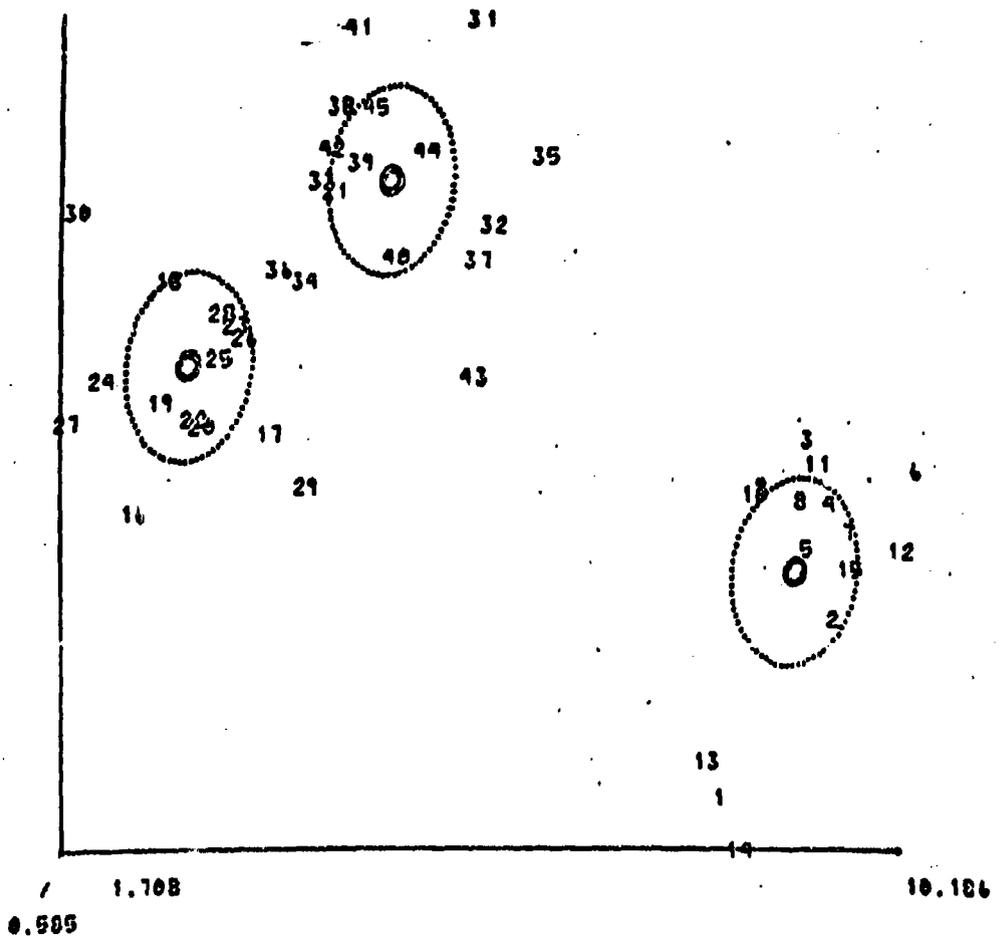
8.529



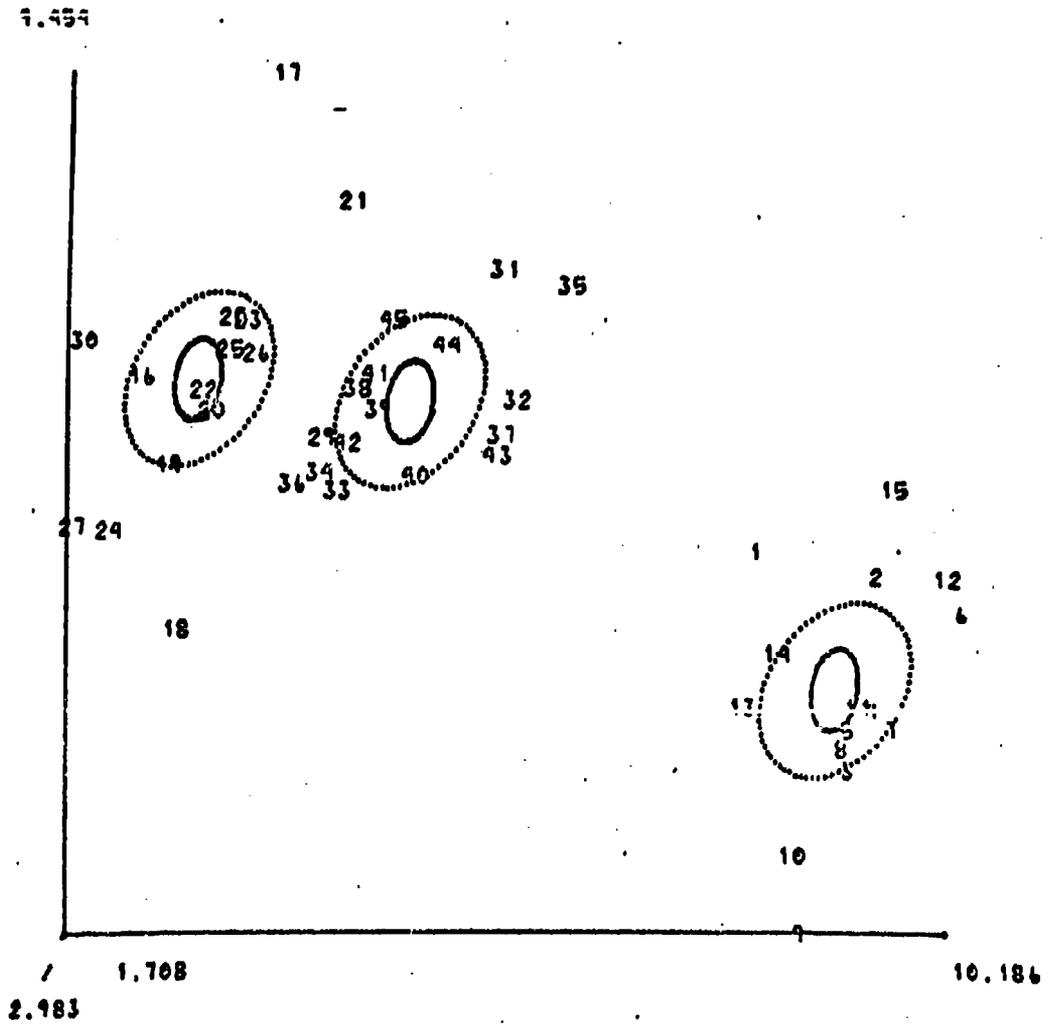
VARIABLE AGAINST VECTOR BELOW, (SIMPLE PLANE NO. 3 SUPERIMPOSED)

1.000 0.0 0.0 0.0

8.521



VARIABLE 4 AGAINST VECTOR BELOW (SIMPLE PLANE NO. 2 SUPERIMPOSED)
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, 1957.
- (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, 1967.
- (11) Cuttman, 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 Francisco and London, 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 CLUSTER Program

DATE 71-257/ZL-41-31

05/300 FORTRAN M

LEVEL 19 (JUNE 70)

```

COMPILER OPTIONS - NAMEC MAIN,OPT=DC,LINECAT=59,SI/E=00000,
SOURCE,B,D,NCL:ST,ADDICK,LCRD,MAP,NCLGIT,NOIO,NDRREF
IMPLICIT REAL*8(I,D)
DOUBLE PRECISION DETAX,DY,ARG,DR,DRN,OF,HALF,YERRN,DLGCM,DETAP
DIMENSION SEIP2500,MSI2750,FMIS200,FSG0,RI,RI2750,MS500,DS200,FFCAN COT
16(1000),FRM(400),KELI(20),MTEP(20),FSET(6),FRMT(40)
DIMENSION DV(20),M(275),JED(156),ESI(56),NLI(50),CL(50),COT,IZ26
1,VAR(20),COR(20),PA(20),FS(20),AC(15),MNS(15),D(15),
COPUR C55(10),JES(12),OS(10),A(50),A(20),M(20),M(20),M(20),M(20)
NCLGOT,NIG(10),NCS(20),AUR(20)
HEADING 14
MSTERS(220)
248 FORPAT(1)
50 FORPAT(13),K(13),A(13),F(0)
51 FORPAT(1),M(1),I(1),O(1),M(1),I(1)
65(5),I(1),F(1)
100 FORPAT(20)
DC 1 I-1,AP
READ(5),FM(1) (R(1),J)-1,NV)
WRITE(14,105) (X(1),J)-1,NV)
105 FORPAT(1),S(16),B(1)
1 CONTINUE
HEADING 14
JJD=0
444 JJD=JJD+1
IF (JJD.EC.4) GC TO 811
WRITE(6,111) JJD
111 FORPAT(1),PASS NO. '121'
IF (JJD.EC.2) ALFCR=AC2
IF (JJD.EC.2) ALFET=AE2
IF (JJD.EC.3) ALFCR=AC3
IF (JJD.EC.3) ALFET=AE3
WRITE(5,51) ALFCR,ALFET
52 FORPAT(1),ALFCR='FS-5',ALFET='F(1)-3'
CALL CER2
GC TO 446
445 CALL COR1
446 CONTINUE
NV=(NV*(NV+1))/2
WRITE(6,110)
110 FORPAT(1),P59 IS UPPER TRI PART OF METRIC USED FOR STANDARDIZATION*
1)
DC 2 I-1,NV
IF (JJD.EC.1) GC TO 447
OSPRG(1)=OSPRG(1)/(1/NV)*MGI
GC TO 449
447 OSPRG(1)=OSPRG(1)/(M-1)
448 CONTINUE
3 WRITE(6,300) I=OSPRG(1)
300 FORPAT(1),MFI='(3)',F(12)=F(12)+F(12)
CALL DMFC(1),M(1),L(10),I(1)
GC 4 I-1,AP

```



```

15N 0119 LPT=LPT
15N 0120 ENSC=L-0
15N 0121 KK=1
15N 0122 KK=KK+1
15N 0123 WSUM=0.0
15A 0124 KK=KK-1
15A 0125 DC 26 J=1,KKK
15N 0126 WSUM=SUM(WIK(ICR(J),LPT))
15N 0127 ENSC=ENSC+L-0+2.0*WSUM
15N 0128 ICR(KK)=LPT
15N 0129 SUMMAX=30-C
15N 0130 CC 29 I=L,NF
15N 0131 CC 29 J=L,NF
15N 0132 IF(I-EG-ICR(J)) GO TO 29
15N 0133 DC 38 J=L,NP
28 CONTINUE
15N 0134 IF(I-EG-TEXT(J)) GO TO 29
38 CONTINUE
15N 0135 DC 31 KK=L,KK
31 CNSUM=CNSUM+WIK(ICR(KK),LPT)
15N 0136 IF(CNSUM-1.5*SUMMAX) GO TO 22
30NSUM=CNSUM
15N 0137 LPT=LPT+1
32 CONTINUE
29 CONTINUE
15N 0138 IF(SUMMAX-LE-0-G1) GO TO 56
15A 0139 PPAR=SUMMAX/SCRT(ENSC)
15N 0140 KKK=KK+1
15N 0141 WPTTE(6,58) KKK,LPT,RMAX
15N 0142 FORVAL('OGRIT',13,' IN CLUSTER IS POINT ',13,*,* CORR IS',F6.4)
15N 0143 ALF=TESTRMAX,KKK
15N 0144 WPTTE(6,57) ALF
15N 0145 FCRVAL('ALF',*,F6.4)
15N 0146 IF(ALF-LE-ALFCR) GO TO 33
15N 0147 IF(R-EG-2) ICL(1,PT)=2
15N 0148 IF(1/2*GT-ALFCR)-AND-(ALF-LE-ALFEXT)) GO TO 34
15N 0149 CC TC 95
33 ICL(1,PT)=1
15N 0150 C(L(PT),KCL)=ALF
15N 0151 GO TO 35
34 IF(1CL(1,PT)-EG-1) GO TO 37
15N 0152 ICL(1,PT)=2
37 CONTINUE
15N 0153 C(L(PT),KCL)=ALF
15N 0154 IE=IE+1
15N 0155 IEXT(IE)=LPT
15N 0156 CC TC 36
15A 0157 KCL=KCL+1
15N 0158 IF(KCL-EG-21) GO TO 95
15N 0159 CC TO 97
95 CONTINUE
15N 0160 DC 02 KK=1,2
15N 0161 JA=10*(KK-1)+1
15N 0162 JC=JA+9
15N 0163 WPTTE(6,45)

```

```

15N 0184
15N 0185
15N 0187
15N 0188
15N 0189
15N 0190
15N 0191
15N 0192
15N 0193
15N 0194
15N 0195
15N 0196
15N 0197
15N 0198
15N 0199
15N 0200
15N 0201
15N 0202
15N 0203
15N 0204
15N 0205
15N 0207
15N 0208
15N 0209
15N 0210
15N 0211
15N 0212
15N 0213
15N 0214
15N 0215
15N 0216
15N 0217
15N 0219
15N 0220
15N 0222
15N 0223
15N 0224
15N 0225
15N 0226
15N 0227
15N 0228
15N 0229
15N 0230
15N 0231
15N 0232
15N 0234
15N 0235
15N 0236
15N 0238
15N 0239
15N 0240
15N 0242

45 FORMAT(1,'PI NO.',32X,'CLUSTERS')
   IFML(EQ.2) GO TO 113
   WRITE(6,112)
112 FORMAT(3X,'NC. 1',4X,'NC. 2',4X,'NC. 3',4X,'NO. 4',4X,
   'NC. 5',4X,'NC. 6',4X,'NO. 7',4X,'NO. 8',4X,
   'NC. 9',4X,'NO. 10')
   GO TO 115
113 WRITE(6,114)
114 FORMAT(3X,'NC. 11',4X,'NC. 12',4X,'NC. 13',4X,'NO. 14',4X,
   'NC. 15',4X,'NC. 16',4X,'NO. 17',4X,'NO. 18',4X,
   'NC. 19',4X,'NC. 20')
115 CONTINUE
46 FCS=1/(C)
   CG=47 I=1,MP
47 WRITE(6,48) I, (CL(I,J),J=JA,JB)
48 FC=MAIL,'IX,E3,6X 'IGIZX(F8=0)
   DC 401 I=1,200
401 ICFI=C
   DC 402 I=1,20
402 NCU(I)=0
   DC 403 J=1,20
   DC 404 I=1,MP
   IF(EL(I,J)GT.9.0)AND.(CL(I,J).LE.ALFCOR) GO TO 405
   GO TO 404
405 NCU(I)=NCC(I)+1
404 CONTINUE
403 CONTINUE
   DO 407 I=1,20
407 NCM(I)=0
   DO 409 K=1,20
   NMAX=1
   DC 406 J=1,20
   DO 408 J=1,20
   IF(LECMGR(I)) GO TO 406
   C=PAGE
   IF(NC(I).LE.NMAX) GO TO 406
   NMAX=NC(I)
   I2=I
406 CONTINUE
   NCR(I)=I2
409 CONTINUE
411 CONTINUE
   J=0
419 J=J+1
   KPI=KPI+1
   IF(NC(NR(I)).LE.2) GO TO 99
   ICT=0
   DO 413 NI=1,MP
   IF(EL(NI,NR(I))GT.9.0) GO TO 414
   GO TO 413
414 ICT=ICT+1
413 CONTINUE

```

```

15N 0243 IF(C1JJ,EG,21,OR,(JJA,EG,31)) GO TO 501
15N 0244 CC 415 JX-1,20
15N 0245 IA,IC,3
15N 0246 IF(C,REJ1,EG,NOR(JX)) GO TO 415
15N 0247 CC 416 I-1,AP
15N 0248 IF(C,CL(I,NOR(JX)),NE,0,01)-AND,(CL(I,NOR(JX)),NE,0,01)IMATC={MATC+1
15N 0249
15N 0250 418 CONTINUE
15N 0251 ICI1=IC1-2
15N 0252 IF(I,IPATC,GT,2)-AND,(IMATC,LE,IC1)) GO TO 417
15N 0253
15N 0254 GO TO 415
15N 0255
15N 0256 IF(C,CL(I,XX,NOR(JX)),LE,ZLFCOR),AND,(CL(I,XX,NOR(JX)),GT,0,01)-AND,
15N 0257 IIC(I,XX),EG,01) IC(I,XX)=KPI
15N 0258
15N 0259 419 CONTINUE
15N 0260
15N 0261
15N 0262
15N 0263
15N 0264
15N 0265
15N 0266
15N 0267
15N 0268
15N 0269
15N 0270
15N 0271
15N 0272
15N 0273
15N 0274
15N 0275
15N 0276
15N 0277
15N 0278
15N 0279
15N 0280
15N 0281
15N 0282
15N 0283
15N 0284
15N 0285
15N 0286
15N 0287
15N 0288
15N 0289
15N 0290
15N 0291
15N 0292
15N 0293
15N 0294
15N 0295
15N 0296
15N 0297
15N 0298
15N 0299
15N 0300
15N 0301
15N 0302
15N 0303
15N 0304
15N 0305
15N 0306

```

```

15N 0307      DSPRGC(I) = CSPROD(I)/(ANNIC-MG)
15N 0308      WRITE (4,306) I,DSPROD(I)
15N 0309      10000 CONTINUE
15N 0310      DC 11010 I=1,NV
15N 0311      11010 VAX(I) = CSPEC(I)
15N 0312      DC 20000 I=1,AG
15N 0313      IF (NIG(I),E,C) GO TC 20000
15N 0314      DC 20000 J=1,NV
15N 0315      CGSCH(I,J) = EGSUM(I,J)/NIG(I)
15N 0316      20000 CONTINUE
15N 0317      DC 20001 I=1,PG
15N 0318      WRITE(6,11020) (DGSUM(I,J),J=1,NV)
15N 0319      WRITE(7,11020) (CSSUM(I,J),J=1,NV)
15N 0320      20001 CONTINUE
15N 0321      11020 FCMPAT(I,X,E,I,J)
15N 0322      CALL CNFSG(CSPRGC,NV,I,E-10,I,ER)
15N 0323      DC 10001 I=1,NP
15N 0324      IF (IC(I),E,C) GO TC 10001
15N 0325      KK = IC(I)
15N 0326      DC 10002 J=1,MV
15N 0327      CCA = XI(I,J)
15N 0328      10002 DVI(J) = OCF - DGSUM(KK,J)
15N 0329      GC TO 10005
15N 0330      10003 DC 10004 J=1,NV
15N 0331      COX = X(I,J)
15N 0332      10004 GVI(J) = OEX - DKSUM(J)
15N 0333      10005 CALL EMOS(EV,I,NV,DSPRGC=1,J,ER)
15N 0334      DC 10007 J=1,NV
15N 0335      XI(I,J) = GVI(J)
15N 0336      10007 CONTINUE
15N 0337      WRITE(15,2) I,X(I,J),J=1,NV)
15N 0338      10001 CONTINUE
15N 0339      NV # INVRNVI<<2
15N 0340      DC 11000 I=1,NV
15N 0341      SPROD(I) = ESPROD(I)
15N 0342      NV = NV**2
15N 0343      REND 15
15N 0344      KFACT = NV
15N 0345      USE WITH SUBROUTINES FACT,WRIP,UTERM,IRSLD
15N 0346      NIN=5
15N 0347      NGUT=6
15N 0348      CALF = 0.0100
15N 0349      LIMITATION=50 VARIABLES, 20 FACTOR:
15N 0350      REAC LEADER CARC
15N 0351      DC 17 KARE = 1.0P
15N 0352      2 FGRAT(I,X,I2,3X,7F10.7)
15N 0353      DC 13 J=1,KFACT
15N 0354      1X 8 2J-1=ONP 6 I0
15N 0355      19 FM(I,X) = SETUP(I)
15N 0356      17 CONTINUE
15N 0357      DC 14 K=1,NF
15N 0358      NZK #0=
15N 0359      DC 14 J=1,KFACT
15N 0360      KX # 2J-1=ONP 6 K
15N 0361      14 MK) = MIN) + FM(K)I=2

```

```

FCAN 000
FCAN006
FCAN 009
FCAN 011

```



```

15N 0479 741 SETUP(JK) = FRIP(J)/SORI(SSI)
15N 0480 WRITE (6,790)
15N 0481 TSO FCPTA11MS,10R,TRANSFORMATION VECTOR TO TRANSECRM RAW DATA TO
      1SIMPLE STRUCTURE)
      CALL PRINSETUP,KFACT=0)
      WFTL (7,11C2) (SETUP)JL=JL-1,KFACT)
      CALL PPRD (PRIM,VARX,Z,LYNY6,LYNY)
      DO 1100 I=1,NV
      K = (L+((L-1)/2)
11000 CORRILL = ZILL/SORIVAR(RJ)
      WRITE (6,11001) (CORRILL)ALL=1,NV)
11004 FCPTA11MS,7R,CORRELATIONS BETWEEN ORIGINAL VARIABLES AND THIS
      VECTOR ARE: //10F12.4)
      DC 11005 LL=1,NP
      INSELLELL=SCU) = 0
      IF (RTEILL)CC=0,0) INSELLELL=NSCU = 1
11005 CONTINUE
11001 IFFNSOLC 1040,1640,1041
1640 WRITE(OUT)16424
1642 FCPTA11MS,21X,25MNC SIMPLE STRUCTURE SOLUT=CNXC
      GC TC 1643
1641 DO 1070 J01,MSCL
      LC 1070 181,J
      SUM # 0.
      DC 1071 K01,KFACT E K
      K1 # 21-16,KFACT E K
      KJ # 21-16,KFACT E K
1071 SUM # SUM # 21,NINXIC=RENTRAC
      IJ = #211,J)
1070 WRITE(OUT)1073
      KRIJJC # SUM
1073 FCPTA11MS,18F,33MCSINMS BETWEEN REFERENCE VECTORS<
      CALL WRTA11MSGL,K
1000 FCPTA11MS,27X,18PMSUMMARY OF RESULTS/K
      APE # INSEL=1/6 E 1
      LE # MCD INSEL=AC
      DO 1002 K01,NAE
      IFFSA=KACC 1004,1604,1606,1606
1004 K05=6
      GC TC 1605
1606 IFFIEX 1004,1604,1604,1610
1610 K06=1E
1608 DC 1612 K061,PRE
1612 K51TRAC E RD E 02KNA=1K
      WRITE(OUT)101K,2KSETZJ,K,J01,R0EC
1614 FCPTA11MS/10H VAR V 12, 8H V 12, 8H
      I V 12, 8H V 12, 8H V 12, 8H
      DC 1616 K061,AMP
      OG 1616 K061,RDE
      SUM # 0.
      DC 1620 K061,FFACT
      K01 # 210-160P E KC
      P1J = (NE-1)E6 # MD1 - 1)KFACT = NG
1620 SUM # SUM # 21,NINXIC=RENTRAC

```

```

FCAN 282
FCAN 283
FCAN 284
FCAN 285
FCAN 286
FCAN 287
FCAN 288
FCAN 289
FCAN 290
FCAN 291
FCAN 292
FCAN 293
FCAN 295
FCAN 296
FCAN 297
FCAN 298
FCAN 299
FCAN 300
FCAN 301
FCAN 302
FCAN 303
FCAN 304
FCAN 305
FCAN 306
FCAN 307
FCAN 308
FCAN 309
FCAN 310
FCAN 311
FCAN 312
FCAN 313
FCAN 314
FCAN 316
FCAN 317
FCAN 318
FCAN 319
FCAN 320

```


LEVEL 15 (JUNE 70)

OS/360 FORTRAN M

DATE 71-257/21.42.06

```
COMPILER OPTIONS - NAME= MAIN,OPT=00,LINE:NT=50,SIZE=0000K,  
SOURCE,BCD,NOLIST,NODECK,LOAD=AP,NOECIT,NOID,NOXREF  
ISN 0002 FUNCTION KD(I,J)  
ISN 0003 IF(I.LT.J) GC TC 1  
ISN 0005 KC=(I*(I-1))/2+J  
ISN 0006 GC TC 95  
ISN 0007 1 KC=(J*(J-1))/2+I  
ISN 0008 99 RETURN  
ISN 0009 END
```

COMPILER OPTIONS - NAME= MAIN,GPT=00,LINECNT=51,SIZE=UCOOK,
SOURCE,BCC,NOLIST,NODECK,LOAD,NAP,NODEIT,NOID,NOXREF

ISN	CHAR	SUBROUTINE FOR WRITING CORRELATIONS, MEANS, AND FACTOR LOADINGS	
ISN 0002	C	WRITE(N,1) 'N=1'	042
ISN 0003	C	WRITE(N,1) 'N=2'	043
ISN 0004	C	WRITE(N,1) 'N=3'	044
ISN 0005	C	WRITE(N,1) 'N=4'	045
ISN 0006	C	WRITE(N,1) 'N=5'	046
ISN 0007	C	WRITE(N,1) 'N=6'	047
ISN 0008	C	WRITE(N,1) 'N=7'	048
ISN 0009	C	WRITE(N,1) 'N=8'	049
ISN 0010	C	WRITE(N,1) 'N=9'	050
ISN 0011	C	WRITE(N,1) 'N=10'	051
ISN 0012	C	WRITE(N,1) 'N=11'	052
ISN 0013	C	WRITE(N,1) 'N=12'	053
ISN 0014	C	WRITE(N,1) 'N=13'	054
ISN 0015	C	WRITE(N,1) 'N=14'	055
ISN 0016	C	WRITE(N,1) 'N=15'	056
ISN 0017	C	WRITE(N,1) 'N=16'	057
ISN 0018	C	WRITE(N,1) 'N=17'	058
ISN 0019	C	WRITE(N,1) 'N=18'	059
ISN 0020	C	WRITE(N,1) 'N=19'	060
ISN 0021	C	WRITE(N,1) 'N=20'	061
ISN 0022	C	WRITE(N,1) 'N=21'	062
ISN 0023	C	WRITE(N,1) 'N=22'	063
ISN 0024	C	WRITE(N,1) 'N=23'	064
ISN 0025	C	WRITE(N,1) 'N=24'	065
ISN 0026	C	WRITE(N,1) 'N=25'	066
ISN 0027	C	WRITE(N,1) 'N=26'	067
ISN 0028	C	WRITE(N,1) 'N=27'	068
ISN 0029	C	WRITE(N,1) 'N=28'	069
ISN 0030	C	WRITE(N,1) 'N=29'	070
ISN 0031	C	WRITE(N,1) 'N=30'	071
ISN 0032	C	WRITE(N,1) 'N=31'	072
ISN 0033	C	WRITE(N,1) 'N=32'	073
ISN 0034	C	WRITE(N,1) 'N=33'	074
ISN 0035	C	WRITE(N,1) 'N=34'	075
ISN 0036	C	WRITE(N,1) 'N=35'	076
ISN 0037	C	WRITE(N,1) 'N=36'	077

NOT REPRODUCIBLE

LEVEL 19 (JUNE 70)

OS/360 FOR RAN H

COMPILER OPTIONS - NAME= MAIN,OPT=0C,LINECNT=59 SIZE=0000K,
SOURCE,BCD,NOLIST,NODECK,LOAD=AP,NODEBIT,NOID,NOXREF

```

SUBROUTINE CCG1
  IMPLICIT REAL*8(D)
  COMMON UCSUM(20,20),DXSUM(20),OSPROD(100),X(50,20),NV,NP,NG,
  IIC(100),MIG(20),ACC(20),NCR(20)
  AC=(NV*NP+I)/2
  DO 1 I=1,NV
  1 DXSUM(I)=0.00
  DO 2 I=1,AC
  2 DSPRCC(I)=0.00
  DO 3 I=1,NV
  DO 4 J=1,NP
  DX=X(I,J)
  4 DXSUM(I)=DXSUM(I)+DX
  DXSUM(I)=DXSUM(I)/CFLOAT(NP)
  WRITE(6,200) I,DXSUM(I)
  200 FORMAT(' MEAN OF VAR(',I2,')= ',F9.4)
  3 CONTINUE
  WRITE(6,300)
  300 FORMAT(' OSFRG IS UPPER TRI PART OF MATRIX OF CORRECTED S.S. AND
  IS.P. FOR NV VARS. STORED IN COLS')
  DO 5 J=1,NV
  DO 6 I=1,J
  KI=K(I,J)
  DO 7 L=1,NP
  DX=X(L,J)
  DX2=X(L,I)
  7 DSPRCC(KI)=DSPRCC(KI)+DXI*DX2
  DSPRCC(KI)=DSPRCC(KI)-DXSUM(I)*DXSUM(L)+DFLOAT(NP)
  WRITE(6,100) KI,DSPRCC(KI)
  100 FORMAT(' DSPRCC(',I3,')= ',F13.4)
  6 CONTINUE
  5 CONTINUE
  RETURN
  END
  ISN C020
  ISN C021
  ISN C022
  ISN C023
  ISN C024
  ISN C025
  ISN C026
  ISN C027
  ISN C028
  ISN C029
  ISN C030
  ISN C031
  ISN C032
  ISN C033
  
```

```

COMPILER OPTIONS - NAME= MAIN,OPT=CO,LINENY=5,SIZE=0000K,
SOURCE,BED,NOLIST,MODEX,LOA,MAP,MODEIT,NOID,NOXREF
SUBROUTINE CORZ
IMPLICIT REAL*8(D)
COMMON DGSUM(20,20),DXSUM(20),DSPROD(500),X(50,20),NY,NP,NS,
1IC(100),NIC(20),ACQ(20),NCR(20)
NC=(NY*(NP+1))/2
DO 2 I=1,AC
2 DSPRCE(I)=0.0C
DO 11 I=1,NG
DO 12 J=1,NY
12 DGSUM(I,J)=C.00
11 CONTINUE
DO 13 P=1,NG
DO 3 I=1,NY
DO 4 J=1,NP
DX=I/J.11
DGSUM(I,J)=DGSUM(I,J)+DX
4 CONTINUE
3 CONTINUE
13 CONTINUE
DO 57 I=1,NY
DO 58 K=1,NG
WRITE(6,155) I,P,DGSUM(I,J)
155 FORMAT(15P) SUP OF OBSERS. FOR VAR. 12, IN GROUP. 12, = ,012.4)
58 CONTINUE
57 CONTINUE
WRITE(6,300)
300 FORMAT(10) DSPROD IS UPPER TRI PART OF MATRIX CF CORRECTED S-S. AND
15.P. FOR NY VARS, STORED BY-COLS-)
DO 5 J=1,NY
DO 6 I=1,J
KI=K(I,J)
DO 7 L=1,PP
17(I0(L),EC=0) GO TO 7
OXL=X(L,J)
OZ2=X(L,I)
DSPRCE(KI)=CSPRD(KI)+OJ+OZ2
7 CONTINUE
OCS=C+OZ
DO 14 L=1,NG
17(I0(L),EC=0) GO TO 23
OOR=OOR+DGSUM(L,J)*OCSUM(L,1)/NIC(1)
GO TO 14
23 NC=NG-1
14 CONTINUE
DSPRCE(KI)=CSPRD(KI)-OOR
WRITE(6,100) KI,DSPRCE(KI)
100 FORMAT(10) ESPRCE(13,*) = ,013.4)
6 CONTINUE
5 CONTINUE
RETURN
END

```

NOT REPRODUCIBLE

DATE 71.257/21.42.25

05/360 FORTRAN M

LEVEL 19 (JUNE 70)

```

COMPILER OPTIONS - NAME= MAIN,OPT=00,LINECHT=58,SIZE=200CK,
SOURCE,BCD,NCLIST,NODECK,LOAD,MIP,NCECIT,NOID,NOXREF
ISN 0002 FUNCTION TEST(EPAK,K)
ISN 0003 IMPLICIT REAL*(EID)
ISN 0004 DOUBLE PRECISION BETAX
ISN 0005 DOUBLE PRECISION YORPA
ISN 0006 DOUBLE PRECISION DLGGM
ISN 0007 CCMPC: DGSUP(20,20),DKSUM(20),DSPROO(40),X(50,20),NV,NP,NG,
      I(1,00),NIG(20),MCO(20),NCR(20)
ISN 0008 CRMAX=KMAX
ISN 0009 CRMAX=1.00-ERRAX**2
ISN 0010 DR=K
ISN 0011 DR=NV
ISN 0012 DR=(CV-1.00)**.500
ISN 0013 DR=NP
ISN 0014 DR=DN-DK*1.00
ISN 0015 DR=1.00*BETAX(CRMAX,CV,.500)
ISN 0016 CRK=CR-1.00
ISN 0017 DLST=BETAX(CPM,DKK,CON)
ISN 0018 TEST=DIEST
ISN 0019 RTURN
ISN 0020 END

```

NOT REPRODUCIBLE

COMPILER OPTIONS - NAME= MAIN.OPT=CG.LIRECNT=58,SIZE=0C00K,
SOURCE=CC.NLIST,NOCHECK,LOAD,NAP,ACCENT,NC10,NOXREF

C DEZAP REQUIRES FUNCTION SUBROUTINES BETAX,CLCOP, AND YCRAX
EQUALE PRECISION FUNCTION METAPROP,DA,ENK
IMPLICIT REAL *80,84

DIMENSION CARG84<<,CFUN84<<
NCT 8 6
C16C,CO

CUPI,CO
IFZCP*DU-07<< 95,51,2C
IFZCP*LE,DIK,CP,IR,LE,12<< GO TO 55

IFZCP*E,07<< GO TO 9L
IFZCP*E,07<< GO TO 9L
IFZCP*E,07<< GO TO 9L

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

DL 9 0
DL 9 0
DL 9 0
DL 9 0

NOT REPRODUCIBLE

```

ISN C068      OMPU # OMP
ISN C069      OMP # 5.00*10PP-DL< C DL
ISN C070      OFUNE # BETAXCMP,DM,CNK - DP
ISN C071      IF=DFUNE < 123.1.40
ISN C072      DU # CMP
ISN C073      CUX # DFUNE
ISN C074      DL # CMFU
ISN C075      DLX # DFUAZ3<
ISN C076      IF=JJ-10< 86.85.195
ISN C077      IF=DECR-GE.C-100*DU-DL<< GO TO 184
ISN C079      OMPU # CMP
ISN C080      OMP # CU - 5.00*DEC.
ISN C081      CFUNE # BETAXCMP,DM,CNK - DP
ISN C082      IF=DFUNE < 41.1.124
ISN C083      CU # CMFU
ISN C084      DLX # DFUAZ3<
ISN C085      DL # CMP
ISN C086      DLX # CFUNE
ISN C087      IF=JJ-10< 86.89.155
ISN C088      DL # CMP
ISN C089      DLX # DFUNE
ISN C090      IF=JJ-10< 86.85.195
ISN C091      DL # CMP
ISN C092      DLX # DFUNE
ISN C093      IF=JJ-10< 86.85.195
ISN C094      BETAP # DMP
ISN C095      RETURN
ISN C096      DRES # CFUNE & DP
ISN C097      WRITESGUT,156< CP,DM,DM,CMP,DRES
ISN C098      FCRMAT=10,5X,43HNO CONVERGENCE IN BETAP IN DOUBLE PRECISION /
              11X,5HINPUT P = 016.10.4H M = 018.10.4H N = 018.10.4H LAST X =
              2016.10.13H FRCDOUES P = 018.10<
              GC TO 1
ISN C099      OMP # DP
ISN C100      GC TO 1
ISN C101      GC TO 1
ISN C102      OMP # DZ
ISN C103      WRITE 3NOUT,101< CP,DM,DM
ISN C104      14M N = 012.4, 28M RESULT HAS BEEN SET TO ZERO <
              GC TO 1
ISN C105      OMP # DU - 30L-CP<<3DU/DNC
ISN C106      GC TO 1
ISN C107      GC TO 1
ISN C108      OMP # DP<<3CU/DNC
ISN C109      GC TO 1
ISN C110      END

```

LEVEL 19 (JUNE 70) OS/360 FORTRAN M

COMPILER OPTIONS - NAME= MAIN,OPT=OC,LINECNT=56,S ZE=000CK,
SOURCE,BCD,NOLIST,NOCHECK,LOAD,N,P,NCEDIT,NOID,NOXREF

```

15N 0002  CCURLE PRECISION FUNCTION YCRMX(DZ)
15N 0003  EXPLICIT REAL*8(C)
15N 0004  DCURLE PRECISION YCRMX
15N 0005  CPI = 39894228041433
15N 0006  DX = CABS(IZ)
15N 0007  IF(CX.GT.3.D0) GO TO 10
15N 0008  DAL = 0.00
15N 0009  CBL = 1.00
15N 0010  CSH = CX
15N 0011  CSM = 1.00
15N 0012  CAN = 0.00
15N 0013  DAN = CFN * 1.00
15N 0014  CAI = -(2.00*CAN - 1.00) *DX*DX
15N 0015  CEI = 4.00*CAN - 1.00
15N 0016  DAL = DBI*CAN + CAI*DAL
15N 0017  DEL = CBI*CBM + CAI * DEL
15N 0018  DAI = DX*DX - CAI
15N 0019  CBI = 2.00 + CBI
15N 0020  CBM = CBI*DAL + CAI*CBM
15N 0021  CSM = CBI*CBM + DAI*CBM
15N 0022  CFA = CAL/CBL
15N 0023  CFB = CBF/CBM
15N 0024  IF(CFB.EQ.0.00) GO TO 20
15N 0025  IF(CABS((CFB-CFA)/CFA).LE.1.0-14) GO TO 20
15N 0026  GO TO 5
15N 0027  CBL = 0.00
15N 0028  CBL = 1.00
15N 0029  CSM = 1.00
15N 0030  CBM = UX
15N 0031  CBI = DX
15N 0032  DAN = 1.00
15N 0033  DFN = DAN + 1.00
15N 0034  CEI = DAN - 1.00
15N 0035  CAC = CBI*CAN + DAI*DAL
15N 0036  OPC = CEI*CBM + CAI*CBL
15N 0037  CFB = CAC/CEC
15N 0038  CAL = DAN
15N 0039  DEL = CBH
15N 0040  CCM = CAC
15N 0041  CSM = DEC
15N 0042  IF(CABS((CFB-CFA)/CFA).LE.1.0-14) GO TO 20
15N 0043  DFA = CFB
15N 0044  GO TO 15
15N 0045  YCRMX = CPI*CFB*DEXP(-DX*CX/2.00)
15N 0046  IF(CX.LE.3.E0) YCRMX = 0.500 - YCRMX
15N 0047  IF(DZ.GT.0.E0) YCRMX = 1.00 - YCRMX
15N 0048  RETURN
15N 0049  END

```

LEVEL 19 (JUNE 7C)

OS/360 FORTRAN M

DATE 71.257/21.42.43

```
CCMPILER OPTIONS - NAME= MAIN,OPT=CG,LINECNT=53,SIZE=0000K,  
SOURCE,BCC,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,NOXREF  
COUBLE PRECISION FUNCTION DLGGM(DX)  
IMPLICIT REAL *8(D)  
DY=DX  
CTERM=1.00  
IF (DX) 1,2  
DLGGM=0.00  
RETURN  
IF (DY-12.00) 3,3,4  
CTERM=CTERM*DY  
CY=DY*1.00  
CCT02  
DLGGM= (CY-.500)* DLGGM(DY)-DY*(12.00*DY)-1.00/(1366.00*DY**3)  
1 1.00/ (1266.00*DY**5) -1.00/ (1680.30*DY**7) 1*.91893853320467300  
2 -DLGGM(DTERM)  
RETURN  
END  
1SA 0002  
1SA 0003  
1SA 0004  
1SA 0005  
1SA 0006  
1SA 0007 1  
1SA 0008  
1SA 0009 2  
1SA 0010 3  
1SA 0011  
1SA 0012 4  
1SA 0013  
1SA 0014  
1SA 0015
```

NOT REPRODUCIBLE

COMPILER OPTIONS - NAME= MAIN,OPT=CO,LINECNT=56,SIZE=COOOR, SOURCE,BCD,NBLIST,NOBECK,LOAD,NAP,NOCEIT,NO10,NOIDREF

```

15A 0002  DOURLR PRECISICA FUNCTIONA BETAXIDX,DM,DN)
15A 0003  IMPLICIT REAL *8 (A,Y,0)
15A 0004  DZ=0.00
15A 0005  CE=1.000
15A 0006  DX = DM - DE
15A 0007  CSUM=Z
15A 0008  IF ( CM.LE. 0.1) GOIC1
15A 0009  IF ( CM.GE. 0.1) GOIC2
15A 0010  IF ( CM.GE. 0.1) GOIC2
15A 0011  IF ( CM.GE. 0.1) GOIC2
15A 0012  IF ( CM.GE. 0.1) GOIC2
15A 0013  IF ( CM.GE. 0.1) GOIC2
15A 0014  IF ( CM.GE. 0.1) GOIC2
15A 0015  IF ( CM.GE. 0.1) GOIC2
15A 0016  IF ( CM.GE. 0.1) GOIC2
15A 0017  IF ( CM.GE. 0.1) GOIC2
15A 0018  IF ( CM.GE. 0.1) GOIC2
15A 0019  IF ( CM.GE. 0.1) GOIC2
15A 0020  IF ( CM.GE. 0.1) GOIC2
15A 0021  IF ( CM.GE. 0.1) GOIC2
15A 0022  IF ( CM.GE. 0.1) GOIC2
15A 0023  IF ( CM.GE. 0.1) GOIC2
15A 0024  IF ( CM.GE. 0.1) GOIC2
15A 0025  IF ( CM.GE. 0.1) GOIC2
15A 0026  IF ( CM.GE. 0.1) GOIC2
15A 0027  IF ( CM.GE. 0.1) GOIC2
15A 0028  IF ( CM.GE. 0.1) GOIC2
15A 0029  IF ( CM.GE. 0.1) GOIC2
15A 0030  IF ( CM.GE. 0.1) GOIC2
15A 0031  IF ( CM.GE. 0.1) GOIC2
15A 0032  IF ( CM.GE. 0.1) GOIC2
15A 0033  IF ( CM.GE. 0.1) GOIC2
15A 0034  IF ( CM.GE. 0.1) GOIC2
15A 0035  IF ( CM.GE. 0.1) GOIC2
15A 0036  IF ( CM.GE. 0.1) GOIC2
15A 0037  IF ( CM.GE. 0.1) GOIC2
15A 0038  IF ( CM.GE. 0.1) GOIC2
15A 0039  IF ( CM.GE. 0.1) GOIC2
15A 0040  IF ( CM.GE. 0.1) GOIC2
15A 0041  IF ( CM.GE. 0.1) GOIC2
15A 0042  IF ( CM.GE. 0.1) GOIC2
15A 0043  IF ( CM.GE. 0.1) GOIC2
15A 0044  IF ( CM.GE. 0.1) GOIC2
15A 0045  IF ( CM.GE. 0.1) GOIC2
15A 0046  IF ( CM.GE. 0.1) GOIC2
15A 0047  IF ( CM.GE. 0.1) GOIC2
15A 0048  IF ( CM.GE. 0.1) GOIC2
15A 0049  IF ( CM.GE. 0.1) GOIC2
15A 0050  IF ( CM.GE. 0.1) GOIC2
15A 0051  IF ( CM.GE. 0.1) GOIC2
15A 0052  IF ( CM.GE. 0.1) GOIC2
15A 0053  IF ( CM.GE. 0.1) GOIC2
15A 0054  IF ( CM.GE. 0.1) GOIC2
15A 0055  IF ( CM.GE. 0.1) GOIC2
15A 0056  IF ( CM.GE. 0.1) GOIC2
15A 0057  IF ( CM.GE. 0.1) GOIC2
15A 0058  IF ( CM.GE. 0.1) GOIC2
15A 0059  IF ( CM.GE. 0.1) GOIC2
15A 0060  IF ( CM.GE. 0.1) GOIC2
15A 0061  IF ( CM.GE. 0.1) GOIC2
15A 0062  IF ( CM.GE. 0.1) GOIC2
15A 0063  IF ( CM.GE. 0.1) GOIC2
15A 0064  IF ( CM.GE. 0.1) GOIC2
15A 0065  IF ( CM.GE. 0.1) GOIC2
15A 0066  IF ( CM.GE. 0.1) GOIC2
15A 0067  IF ( CM.GE. 0.1) GOIC2
15A 0068  IF ( CM.GE. 0.1) GOIC2

```

NOT REPRODUCIBLE


```

ISN 0131
ISN 0132
ISN 0133
ISN 0134
ISN 0135
ISN 0136
ISN 0137
ISN 0138
ISN 0139

4 DA = DNT
  GC TO 7
  WRITE(G,LOG) CP,CM,CX
  FORMAT(///SX, 4CHERRC IA INPUT PARAMETER BETAX SET TO 0. ,
1 /SX, 2H=, D2G.S,2HN=,D2G.S,2HX=.D20.8)
  BETAX=CZ
  RETURN
3 BETAX=DE
  RETURN
2 RETURN
  ENC

```

APPENDIX B

Source Listings for ELLIPSE Program

FORTRAN IV G LEVEL 19

DATE = 71257

MAIN

14/31/15

```

0001      C THIS IS THE MAIN PROGRAM
          COMMON GDSZ,GDSY,GTEXT,CPOINT,GDSE,GDSEI,X(5,8),NV,MP,NGG,NSQL,
          2RC,CLMEAN(5,8),A(36,11),XMAX(8),XMIN(8),XLO,XLO,XYHI,YNDET(5),
          2R(16),KWRKNG(16),GINPUT,RINPUT(8),SUC(10,8),INNSQL(50,2),IGWRK(50
          3),NIG(5),DSEPRDD(36),IG(50),DCCUNT(5,8),DXSUH(1),A,ARKNG(40)
          CALL CALC
          CALL EXIBIT
          STOP
          END
0002
0003
0004
0005

```

NOT REPRODUCIBLE

14/31/24

DATE = 71257

KD

FORTRAN IV G LEVEL 19

```
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
```

NOT REPRODUCIBLE

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

```

0001 SUMROUTINE CALC
0002 COMMON CDSX, CDSY, CTEXT, CPRINT, CDSR, K(50, 1), NV, NP, KGG, NSOL,
      LAC, CLKAR(5, 8), A(136, 1), X(19), XMIN(8), XLD, YLD, X(1, Y), YDTE(5),
      Z(1, 10), XWANG(1, 8), CINPUT(8), SUG(10, 8), INNSCL(50, 2), ICKOR(50
      3), NIG(5), OS, P(36), I(50), OSUM(5, 8), OSUM(8), XWANG(40)
      READ (9, 2) NP, NV, KGG, NSOL
      NVV = (NV*(NP+1))/2
      2 FORMAT (4(1A, 13))
0003 30 14 1=1,NG
0004 14 READ (5, 451) (CLNEAM(I, J), J=1, NV)
0005 451 FORMAT (14X, 6F7.3)
0006 30 114 1=1, NSOL
0007 114 READ (5, 451) (SUG(I, J), J=1, NV)
0008 165C FORMAT (13, 12, 11, 12, 6F7.3)
0009 2) 12 1=1, NP
0010 READ (10, 1659) INNSCL(1, 2), I(1), INNSOL(1, 1), X(1), J(1), NV)
0011 10 CONTINUE
0012 13 20 225 1=1, NG
0013 425 NIG(1) = 0
0014 30 422 1=1, NG
0015 20 423 J=1, NP
0016 15 (164), 50, 1) NIG(1) = NIG(1) + 1
0017 423 CONTINUE
0018 422 CONTINUE
0019 NVAL = 0
0020 30 420 1=1, NG
0021 420 NVAL = NVAL + NIG(1)
0022 30 420 1=1, NP
0023 ICKOR(1) = ICK(1)
0024 420 CONTINUE
0025 NCG = NG
0026 CALL CORZ
0027 30 434 1=1, NVV
0028 434 OSPRC(1) = OSPRC(1)/NVAL(NG)
0029 CALL SINVS(PSQD, NVAL, 7, IEN)
0030 412: 1=1, 100) ICR
0031 1661 FORMAT (18, *ERROR CODE AFTER CALL TO SINV IS*, I3)
0032 30 435 1=1, NVV
0033 411(1) = OSPRC(1)
0034 435 CONTINUE
0035 15 INNSCL(50, 0) GC FC 450
0036 30 425 1=1, NVV
0037 OSUM(1) = G+0
0038 30 435 J=1, NP
0039 456 DCOR(1) = OSUM(1) + X(J, 1)
0040 456 DCOR(1) = OSUM(1)/NP
0041 455 CONTINUE
0042 30 426 1=1, NSOL
0043 30 427 J=1, NP
0044 ICKOR(1) = 0
0045 154 = ICK(10)
0046 15 (12), 1, 1) / 15 = (10-1)
0047 427 CONTINUE
0048 432 NIG(1) = 0

```

NOT REPRODUCIBLE

FORTRAN IV 6 LEVEL 19

CALC

DATE = 71257

4/31/45

```
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 SINVD(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 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
```

NOT REPRODUCIBLE

FORTRAN IV G LEVEL 19

CORIS

DATE = 71257

14/32/73

```

0001 SUBROUTINE CORIS
0002 COMMON GCSX,GDSY,GTEXT,GPUINT,GDSE,GDSER,X(50,8),NV,NP,NGG,NSQL,
0003     1NG,CLMEAN(5,8),A(36,11),XMAX(8),XMIN(8),XLO,YLO,XI,I,YHI,NDET(5),
0004     2R(16),RWRKNG(16),GINPUT,RINPUT(8),SUG(10,8),INNSCL(50,2),IGWRK(50
0005     3I,NIG(5),DSPROD(36),IG(50),DGSUM(5,8),DXSUM(8),AWRKNG(40)
0006     DO 10 I=1,NV
0007     DO 11 J=1,I
0008     KI = K0(I,J)
0009     DSPROD(KI) = 0.0
0010     DO 12 L=1,NP
0011     IF (IGWRK(L).EQ.0) GO TO 12
0012     IF (IG(L).EQ.0) GO TO 7
0013     K = IG(L)
0014     DX1 = X(L,J) - CLMEAN(K,J)
0015     DX2 = X(L,I) - CLMEAN(K,I)
0016     DSPROD(KI) = DSPROD(KI) + DX1*DX2
0017     GO TO 12
0018     7 DX1 = X(L,J) - DXSUM(J)
0019     DX2 = X(L,I) - DXSUM(I)
0020     DSPROD(KI) = DSPROD(KI) + DX1*DX2
0021     12 CONTINUE
0022     WRITE (6,100) KI,DSPROD(KI)
0023     100 FORMAT(' DSPROD(',I3,')=',F13.4)
0024     11 CONTINUE
0025     10 CONTINUE
0026     RETURN
0027     END

```

NOT REPRODUCIBLE

FORTRAN IV G LEVEL 19 CON: DATE = 7/257 14/32/11

```

0001 SURROUTINE CORZ
0002 COMMON CDSX, CDSY, GTEXT, ZPOINT, CDSE, CDSER, XI(50,8), NY, NP, MCG, NSOL,
      1 NG, CLP, ANIS(8), AIB(5,11), XMAX(11), XMIN(8), XLO, XLO, XHI, YHI, YDET(5),
      2 Z(10), MARKNG(15), GINPUT, INPUT(8), SUC(10,8), INNSCL(50,21), ICORR(150)
      3, NIG(5), DSPROD(136), ICI(50), DCSUM(5,8), DCSUM(8), DCSUM(16), ARRANGE(40)
      4 NG=(NY*(NV+1))/Z
      5 DO 2 I=1,NC
      6 2 DSPROD(I)=0,DC
      7 DO 11 I=1,NGC
      8 DO 12 J=1,NV
      9 12 DCSUM(I,J)=0,DC
      10 11 CONTINUE
      11 DO 13 K=1,NGC
      12 DO 3 I=1,NV
      13 DO 4 J=1,MP
      14 3 4 X(I,J,1)
      15 IF (ICORR(K),I)=DCSUM(K,1)+DC
      16 DCSUM(K,1)=DCSUM(K,1)+DC
      17 4 CONTINUE
      18 3 CONTINUE
      19 DO 57 I=1,NV
      20 DO 54 K=1,NGC
      21 54 57 I=1,NV
      22 57 54 K=1,NGC
      23 WRITE(6,300)
      24 300 FUM=11, DSPROD IS UPPER TRI PART OF MATRIX OF CORRECTED S.S. AND
      25 15-P. FOR NV VARS, STORED BY COLS)
      26 59 CONTINUE
      27 57 CONTINUE
      28 54 57 I=1,NV
      29 DO 6 I=1,7J
      30 KI=KOT(I,J)
      31 DO 7 L=1,MP
      32 IF (ICORR(KL),EG,0) GO TO 7
      33 DAI=KIL,JI
      34 DI2=XIL,11
      35 DSPROD(KI)=DSPROD(KI)+CAL*DMZ
      36 7 CONTINUE
      37 DCLP=0,DC
      38 DO 14 L=1,NGC
      39 IF (NIG(L),EG,0) GO TO 23
      40 DCLP=DCOR+DCSUM(L,J)+DCSUM(L,11)/NIG(L)
      41 GO TO 14
      42 23 NGC = NGC-1
      43 14 CONTINUE
      44 DSPROD(KI)=DSPROD(KI)-CCDM
      45 WRITE(6,100) A1, DSPROD(KI)
      46 100 FORMAT(' DSPROD(', I3, ') = ', F13.4)
      47 6 CONTINUE
      48 5 CONTINUE
      49 RETURN
      50 END

```

NOT REPRODUCIBLE

FORTRAN IV C LEVEL 10 EXHIBIT DATE = 71257 14/32/71

```

0001 SUBROUTINE EXHIBIT
0002 COMMON CCSX,CCSY,CTEXT,CPOINT,CCSE,CCSER,K(50,3),MV,MP,KGC,MSDL,
      INGL,MEANIS,BI,AL(3,3),X(3),XMIN(3),XMAX(3),ALGO,YLO,XHI,YHI,MOET(5),
      ZR(10),RWRK(16),GINPUT,INPUT(8),SLC(10,3),INKSDL(50,2),IC-OR(4,9)
0003 31,MI(5),DSPR(3,3),IC(5),DZSUM(5,3),DESUM(8),A-PR(4,4)
0004 CALL DISPLA(CCSX,CCSY,CPOINT,CPOINT,CCSE,CCSER,GINPUT)
0005 WRITE (6,1654)
0006 1654 FORMAT('DISPLA SUCCESSFUL')
0007 1 = LIGHS(1,1),MV,27-27,31)
0008 CALL MESSAGES(CTEXT,CCSX,CCSY,CPOINT,CCSE,CCSER,GINPUT)
0009 CALL MESSAGE
0010 WRITE (4,1655)
0011 1655 FORMAT('FOLLOWING ANY DISPLAY OF PROJECTIONS PRESS')
0012 1=KEY 25 TO CHANGE THE VECTOR FORMING ONE OF THE AXES'
0013 2=KEY 31 TO CHANGE THE VARIABLE FORMING ANOTHER AXIS.'
0014 3=ON PRESS ANY KEY TO GO ON')
0015 CALL MESSAGES(CTEXT,CCSX,CCSY,CPOINT,CCSE,CCSER,GINPUT)
0016 LP = PLOT(CTEXT)
0017 LP = DETAIN(MEET)
0018 LP = UNPLOT(CTEXT)
0019 CALL MESSAGES(CTEXT,CCSX,CCSY,CPOINT,CCSE,CCSER,GINPUT)
0020 N(1) = 1
0021 KGC = 0
0022 20 CALL KEYIN (NAXIS,INTEST)
0023 IF (INXIS.EQ.JEIGN TO 40
0024 CALL ECLPSE (NAXIS)
0025 10 KA = DETAIN(MEET)
0026 KA = 20R(1,4)
0027 17 IF (INX.EQ.30) GO TO 40
0028 INTEST = KA
0029 IF (INX.EQ.29) GO TO 60
0030 IF (INX.EQ.31) GO TO 60
0031 IF (INX.EQ.NSDL) GO TO 61
0032 LP = UNPLOT(CTEXT)
0033 CALL MESSAGES(CTEXT,CCSX,CCSY,CPOINT,CCSE,CCSER,GINPUT)
0034 62 FORMAT('PLICAROR OR SINGULAR SOLUTION - PRESS ANOTHER KEY')
0035 1=IF TRAPPED,PRESS KEY 20')
0036 LP = PLOT(CTEXT)
0037 GO TO 16
0038 LP = UNPLOT(CCSX,CCSY,CPOINT,CTEXT)
0039 CALL MESSAGES(CCSX,CCSY,CPOINT,CCSE,CCSER,GINPUT)
0040 LP = UNPLOT(CSER)
0041 CALL MESSAGES(CTEXT,CCSX,CCSY,CPOINT,CCSE,CCSER,GINPUT)
0042 GO TO 20
0043 61 CALL HELPSE (KA,NAXIS)
0044 IF (INX.EQ.-15) GO TO 103
0045 KA = 15
0046 GO TO 17
0047 KK = DETAIN(MEET)
0048 KK = MOET(4)
0049 LP = UNPLOT(CSER)
0050 CALL MESSAGES(CTEXT,CCSX,CCSY,CPOINT,CCSE,CCSER,GINPUT)
0051 GO TO 17
0052 40 CALL BLANK

```

NOT REPRODUCIBLE

FURIRAN IV G LEVEL 19

EXIBIT

DATE = 71257

14/32/21

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

FORTRAN IV G LEVEL 19

HFSSGE

DATE = 71257

14/12/29

```

0001 SUBROUTINE MESSAGE
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(15),RASKNG(16),GINPUT,RINPUT(S),SUS(10,8),INNSCL(50,2),IGWORN(50
      3),NIG(5),DSPROD(36),IG(50),DGSUM(5,8),DXSUM(8),AARKNG(40)
0004 WRITE (4,460)
460 FORMAT('THIS PROGRAM DISPLAYS CLUSTERS BY PROJECTING THEM ON')
      1'VARIOUS 2-DIMENSIONAL SUBSPACES.SIMPLE STRUCTURE'
      2'POSITIONS CAN ALSO BE INDICATED.REFER TO YOUR'
      3'INSTRUCTION CHART.HAVE YOU ENTERED ALL NECESSARY'
      4'PODATA VIA 'GENERATOR'?
      5'PC',10X,'THE BOTTOM ROW OF THE PROGRAM '
      6'FUNCTION KEYS IS LIT UP.THEY WILL BE USED'
      7'POAS DIRECTED.THE DARK ONE,KEY NO. 30,IS'
      8'THE PANIC BUTTON.IT WILL RETURN YOU TO THE '
      9'MONITOR.'/PLAIN CASE OF PANIC PRESS KEY 30'
      1'NOW PRESS ANY KEY TO GET STARTED')
      CALL RKFMT(GTEXT)
      LP = PLCT(GTEXT)
      WRITE (6,1653) LP
1653 FORMAT('LP AFTER FIRST MESSAGE IS',I4)
      I = DETAIN(NDET)
      I = UNPLCT(GTEXT)
      CALL RESET(GTEXT,GCSX,GDSY,GPOINT,GDSE,GDSER,GINPUT)
      RETURN
      END
0005
0006
0007
0008
0009
0010
0011
0012
0013

```

NOT REPRODUCIBLE

FORTRAN IV G LEVEL 19

KEYIN

DATE = 71257

14/12/38

```
0039 NGG = 1
0040 CALL RECURS
0041 CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDUSER,GINPUT)
0042 101 WRITE (4,1558)
0043 1658 FORMAT('PL PRESS ONE KEY CORRESPONDING TO THE AXIS.'/
             '1'P - VARIABLE YOU WISH TO SELECT, OR 3G IF.'/
             '2'P YOU WISH TO STOP,IF YOU WISH TO MAKE.'/
             '3'P CHANGES IN YOUR VECTOR PRESS KEY 28.')
0044 CALL WRKFM$(GTEXT)
0045 LP = PLCT(GTEXT)
0046 60 NAXIS = DETAIN(NDET)
0047 NAXIS = ADET(4)
0048 IF (NAXIS.EQ.30) RETURN
0049 IF (NAXIS.EQ.28) GO TO 59
0050 IF (NAXIS.LE.NV) GO TO 54
0051 LP = UNPLCT(GTEXT)
0052 CALL PESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDUSER,GINPUT)
0053 WRITE(4,52)
0054 52 FORMAT('PL ERROR - PRESS PROPER KEY')
0055 CALL WRKFM$(GTEXT)
0056 LP = PLCT(GTEXT)
0057 GO TO 60
0058 54 LP = UNPLCT(GTEXT)
0059 CALL RESET(GTEXT,GDSX,GDSY,GPOINT,GDSE,GDUSER,GINPUT)
0060 RETURN
0061 END
```

NOT REPRODUCIBLE

15/32/68

DATE = 71257

FLUPEE

FORTRAN IV C LEVEL 19

```

0C54 CALL LIME(GD5A,XMAX(AXIS1),VXIN1)
0C55 LP = PLOT(GD5X)
0C56 CALL PLACE(GD5Y,XLO,0.5*(VX1+VMAX))
0C57 WRITE (4,120) VMAX
0C58 FORMAT(9F9.3)
0C59 CALL WAFPTS(GD5Y)
0C60 CALL PLACE(GD5Y,XLO,(2.0*ALO+VXIN1)/3.)
0C61 WRITE (4,120) VXIN1
0C62 CALL WAFPTS(GD5Y)
0C63 XLX = (XLO+VXIN1)/3.
0C64 VLA = (VLS-2.0*VXIN1)/3.
0C65 CALL PLACE(GD5Y,XLA,VLA)
0C66 WRITE (4,121) XMIN(AXIS1)
0C67 FORMAT(9F9.3)
0C68 CALL WAFPTS(GD5Y)
0C69 XLAX = (XLO+XMAX(AXIS1)-2.0*AX1)/7.
0C70 CALL PLACE(GD5Y,XLAX,XLX)
0C71 WRITE (4,120) XMAX(AXIS1)
0C72 CALL WAFPTS(GD5Y)
0C73 LP = PLOT(GD5Y)
0C74 DO 110 K=1,NP
0C75 CALL PLACE(POINT,UNIT,VXIN1)
0C76 WRITE (4,111) INXSEL(K,2)
0C77 FORMAT (9F9.3,12)
0C78 CALL WAFPTS(GD5Y)
0C79 CONTINUE
0C80 LP = PLOT(POINT)
0C81 ANGLE = 1.0
0C82 DEN = (SII-SJJ)**2 + 4.0*SIJ**2
0C83 IF (DEN.EQ.0.0) ANGLE = SORT(C.5-SORT(C.25-SIJ**2/DEN))
0C84 AXIS1 = SII*ANGLE**2+SJJ*(1.-ANGLE**2)**2+ANGLE*SORT(1.-ANGLE**2)**2
151J
0C85 IF (ABS(1.E-0.0) GO TO 213
0C86 AXIS1 = 1./50*(AXIS1)
0C87 AXIS2 = 11. - ANGLE**2)**2)**2+ANGLE**2**SJJ
1-2.0*ANGLE*SORT(1.-ANGLE**2)**2)**2
0C88 IF (ABS(1.E-0.0) GO TO 213
0C89 AXIS2 = 1./50*(AXIS2)
0C90 SIG = SIGN(1.,(AXIS2-AXIS1)**51J)
0C91 C = 3.14159/180.
0C92 THETA = 5.0
0C93 DO 31 K = 1,NV
0C94 DO 31 KK = 1,NG
0C95 ABRANG ((A-LI)*NG*KK) = CL*WAK (KK,K)
0C96 CONTINUE
0C97 CALL DPHU (A,ABRANG,8,RBRANG,NG,NV,2)
0C98 DO 150 K=1,NG
0C99 DO 150 KK=1,72
100 RADIUS = THETA**K
101 ELX = AXIS1-COS(SINRADIJ)
102 ELY = AXIS2+SINRADIJ
103 ELX(K,KK) = ABRANG(K) + ELX*ANGLE**2)**2)**2
104 ELY(K,KK) = ABRANG(K) + ELY*ANGLE**2)**2)**2
150 CONTINUE
DO 100 KK=1,72
DO 100 K=1,NG

```

NOT REPRODUCIBLE

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

```
0051 RELPSY(K, KK) = RVARKNG(NG+X) + ELX * SQRT(1. - ANGLE**2) * SIG * EL Y * ANGLE
0052 160 CONTINUE
0053 00 170 K=1, NG
0054 00 170 KK=1, 72
0055 CALL POINT(GDUSER, RELPSX(K, KK), RELPSY(K, KK))
0056 170 CONTINUE
0057 LP = PLCT(GDUSER)
0058 RETURN
0059 211 N1 = -15
0060 RETURN
0061 END
```