HOW TO USE KYST

A VERY FLEXIBLE PROGRAM TO DO MULTIDIMENSIONAL SCALING AND UNFOLDING

Joseph B. Kruskal
Bell Laboratories
Murray Hill, N.J.

Forrest W. Young
Thurstone Psychometric Laboratory
University of North Carolina
Chapel Hill, N.C.

Judith B. Seery
Bell Laboratories
Murray Hill, N.J.

KYST is an extremely flexible and portable computer program for multidimensional scaling and unfolding. It represents a merger of M-D-SCAL 5M and TORSCA 9, including the best features of both, as well as some new features of interest. The name, pronounced "kissed", is formed from the initials Kruskal, Young, Shepard, and Torgerson. This instruction manual tells not only how to use KYST, but includes helpful information about how to get it working at a computer center, and other useful material.

KYST includes the powerful initial configuration procedure from TORSCA as well as the very helpful practice of rotating solutions to principal components. KYST incorporates the great generality of M-D-SCAL 5M, as well as M-D-SCAL's easy-to-use input procedure.

New features include improved portability, more readable printer plotting, an easy way to transform the data prior to scaling, an easy way to introduce weights as functions of the data, and substantial economy of computer memory by complicated re-use of arrays.

CONTENTS

1. Introduction 4
2. Input-Preliminary Example 5
3. What KYST Does.................... 6
4. Output....................9
5. Options and Control Phrases.................... 9
6. Basic Control Phrases....................11
7. Control Phrases for Data and Weights.................... 11
8. Data and Weights Decks 15
10. Control Phrases to Choose Analysis Type 21
11. Phrases to Control Termination and Convergence 23
12. Control Phrases for Output 25
13. Some Ways to Use KYST 27
14. User Limitations and Computing Times 29
15. Some Precautions: Number of Dimensions, Convergence, and Local Minima 29
16. The Routines Which Make Up KYST 31
17. Useful Information for Getting KYST Running at a Computer Center 32
References 35
Index of Control Phrases....................36

Figures: 1 5
2 18
3 34

1. Introduction

Multidimensional scaling is a statistical technique which is standard in psychology and is spreading to other fields. Briefly speaking, it constructs a configuration of points in space from information about distances between the points. The literature on the subject includes several books and hundreds of papers. Recent textbooks on psychometrics generally include material on this topic, and some recent texts on statistics do also. While this instruction manual assumes familiarity with the basic ideas of scaling, it does include some brief exposition as a refresher, and to connect the concepts with the program.

Unfolding can be thought of as a kind of scaling in which there are two kinds of points (often called "subject" points and "stimulus" points), such that the information available compares distances radiating from a single subject point at a time to the various stimulus points. While KYST may be used to do pure unfolding, difficulties are not unusual. For example, at the most meaningful solution stress may reach a merely local minimum, and it is not unusual for the program to reach a non-meaningful minimum. To be recommended for use with KYST are many procedures intermediate between unfolding and ordinary scaling, which use two or more sets of data, only one of which is of the unfolding type. Even where the only direct data available are of the unfolding type, it is generally possible and desirable to derive secondary data of another type for supplementary use.

KYST is extremely general because of the many options it contains. In addition to many types of metric and non-metric scaling and unfolding, many similar useful models may be solved which do not bear separate names.

KYST is a merger of M-D-SCAL 5M and TORSCA 9, including the best features of both, as well as several new features. The powerful procedure used by TORSCA to obtain an initial configuration is included in KYST. While this initial configuration works very well in the classical scaling situations, its value in other KYST applications such as unfolding is not always clear.

We believe that it should be quite simple to get KYST running on most computer systems where FORTRAN IV is available. Both M-D-SCAL and TORSCA are widely used, and the main obstacle to portability in M-D-SCAL has been removed from KYST. Useful details are provided in a later section. The considerable printer plotting done by KYST includes several improvements, such as grid labelling in round numbers for ease of interpretation.

In order to reduce the need for preliminary programming to manipulate the data, easy ways to transform the data and to produce weights have been included in KYST.

Substantial economy of memory space has been achieved by a rather intricate re-use of arrays. Dynamic allocation of storage, not yet a portable reality, would have been a more satisfactory solution. Unfortunately, this means that those who wish to modify the program may find it difficult to change array sizes safely. However, a diagram of storage use is included for those who wish to do so.

We welcome feedback from the users of KYST, such as applications of special interest, and novel but useful ways of using the program. We also welcome information about any errors which may remain in the program, especially if accompanied by sufficient information to permit tracking them down, and still more especially if they have already been tracked down and a correction prepared.

2. Input-Preliminary Example

Figure 1 shows the complete input to KYST sufficient to perform one simple example of multidimensional scaling. See Figure 4 at the end of the paper for another example.

```
                Card
               number  Each line is one card          Description

                  1 TORSCA                            Control Card
                  2 PRE-ITERATIONS=3                  Control Card
                  3 DIMMAX=3,DIMMIN=1                 Control Card
                  4 COORDINATES=ROTATE                Control Card
                  5 CARDS                             Control Card
                  6 ITERATIONS=50                     Control Card

                  7 DATA, REGRESSION=DESCENDING       Control Card
Data              8 JOHN SMITH'S CONFUSION MATRIX     Title Card
deck.             9 5 1 1                             Parameter Card
Do not            10 5F5.3                            Format Card
separate          11 9.3 0.57 6.1 2.0 1.5             Data
or re-            12 1.0 8.5 5.5 3.3 4.0              Data
arrange           13 2.1 5.3 7.3 3.9 1.1              Data
these             14 5.0 6.2 1.6 8.9 2.8              Data
cards.            15 1.3 4.1 1.0 2.5 8.8              Data

                  16 COMPUTE                          Control Card
                  17 STOP                             Control Card
```

Figure 1

Note that the card number and description are for reference purposes only. They are not punched on the cards.

In this example, the first six cards happen to be control cards. They are used in this case to give the values of several parameters. Card 1 indicates that the initial configuration is to be generated using the Young-Torgerson procedure. Card 2 states that a maximum of 3 iterations are to be used in this procedure. Card 3 indicates that the scaling is to be done from a "dimension maximum" of three down to a "dimension minimum" of one; that is, separate computations will be performed successively, in 3-dimensional space, in 2-dimensional space, and in 1-dimensional space. Card 4 specifies that the final configuration for each dimensionality should be rotated to principal components. Card 5 indicates that the configuration which results from each scaling should be punched out on cards. Card 6 states that no more than 50 iterations should be performed in any single scaling, even if the stress has not yet reached its minimum value, that is, even if the configuration has not converged to its final position by then. This is a safety precaution, to control computing time. Whenever a scaling is terminated, regardless of whether convergence is complete or not, sufficient output is printed to permit you to start a new computation from where the last one left off. If desired, cards may be punched out to make this restart easier.

Cards 7 through 15 in the example are called a data deck. They must not be separated from each other, nor rearranged internally. However the data deck, treated as an indivisible unit, and many control cards may be freely rearranged: the order in which they occur makes no difference. There are some restrictions however.

Card 7, which is the first card of the data deck, is also a control card, and indeed contains the signal by which the program knows that the succeeding cards should not be read as control cards. Any control card which has the word **DATA** on it must be immediately followed by the remaining cards of a data deck.

The phrase **REGRESSION = DESCENDING** indicates that the regression of distances on data values will be a "monotone descending" regression. This means two things. First, the scaling will be nonmetric, because monotone regression is inherently a nonmetric kind of procedure, as opposed to polynomial regression, which is also possible in KYST. Second, because the regression is descending, small data values will correspond to large distances and hence to objects which are very different from one another. Thus descending regression is appropriate for similarities, and ascending regression is appropriate for dissimilarities.

The internal arrangement of the data deck is rigidly specified unlike the arrangement of individual control cards . The second card of the data deck must be a title card. Its text will be printed and punched out where appropriate, as a title for the data.

The third card of the data deck indicates the number of rows and columns of the matrix namely, 5 , the number of values provided for each matrix entry namely, 1, and the number of such matrices included in the data deck namely, 1.

The fourth card of the data deck must give a format according to the rules for FORTRAN format statements. This format will be used by the program to read a single row of the matrix. For computing machines which handle integers differently from floating-point numbers, a floating-point format must be used.

The succeeding cards of the data deck like cards 11 through 15 must contain the data matrix itself, punched according to the format you have provided. Note that each row of the matrix must begin on a new card. However, several cards may be used for a single row.

A control card with the word **COMPUTE** causes actual computation to start. Everything prior to the **COMPUTE** card constitutes input of data, parameter values, and option choices. All such input relevant to a given computation must occur before the corresponding **COMPUTE** card. This is one exception to the free rearrangeability of control cards.

It is sometimes convenient to perform several different and possibly unrelated scaling computations on a single computer run. For this reason, the program is ready to receive new control cards and new data after it finishes the computation initiated by a **COMPUTE** card. After the last computation, there must be a **STOP** card to signal the end of the computer run.

3. What KYST Does

Only the briefest description of what KYST does can be given here. For a fuller description of the ideas, consult [Kruskal 1964a and 1964b] and [Young 1972] .

KYST places N points in a space of dimension `LDIM` so as to minimize `STRESS`, which measures the "badness-of-fit" between the configuration of points and the data. It finds the minimizing configuration by starting with some configuration perhaps found by the classical scaling procedure of Torgerson , and moving all the points a bit to decrease the stress, then iterating this procedure over and over again until the stopping criterion is reached. Typically, from 15 to 50 iterations may be required. Technically speaking, KYST uses the iterative numerical method of gradients, the method of steepest descent, with a step-size procedure based primarily on the angles between successive gradients.

Let the data values be listed in any order and called `DATA(1)` through `DATA(MM)`. In John Smith's confusion matrix above, $MM = 25$. For a square matrix of size N we have $MM = N^2$ if the diagonal entries are to be included or $MM = N*(N-1)$ if not. For a half matrix $MM = N*(N+1)/2$ or $MM = N*(N-1)/2$ depending on whether the diagonal entries are included or not. Each entry in the matrix may be represented by several distinct data values. If each entry is represented by `NREPL1` data values, then MM is `NREPL1` times as large as it would otherwise be. If some of the data values are missing, MM is appropriately reduced.

For any given configuration of N points (it doesn't make any difference where the configuration comes from), let the distances between pairs of points be calculated and listed in `DIST(1)` to `DIST(MM)` in corresponding positions to those in the data list. If one matrix entry has several data values, then the self same distance will appear in the distance list several times. If some other matrix entry has no data values (missing data situation, for example), then the corresponding distance never appears on the distance list. Corresponding to diagonal entries of the matrix are distances of zero, of course, and the distances corresponding to (i,j) and (j,i) entries of the matrix are of course the same automatically .

Now perform a least-squares regression of `DIST` on `DATA`. For nonmetric scaling, monotone ascending regression should be used for dissimilarities, and monotone descending regression for similarities. For metric scaling, this regression can be linear or it can be polynomial up to 4 degree . Of course, the regression yields the polynomial coefficients. By writing a very simple FORTRAN subroutine, still other types of regression can be accommodated.

Let the values of the regression function be listed in `DHAT(1)` to `DHAT(MM)`. Let `DBAR` be the arithmetic average of the `DIST` values. Then:

$$STRESS = \sqrt{\frac{\sum_{m=1}^{MM}\left(DIST(m) - DHAT(m)\right)^2}{\sum_{m=1}^{MM}\left(DIST(m) - d_0\right)^2}}$$

where:

$$d_0 = \begin{cases} 0 & \text{for stress formula 1} \\ DBAR & \text{for stress formula 2} \end{cases}$$

It is this quantity that is minimized by moving the points of the configuration around.

The two formulas will often yield very similar configurations. In using KYST to do multidimensional unfolding, as explained later, the use of Formula 2 is in theory vital. However, Formula 2 sometimes has problems due to local maxima which cannot occur with Formula 1.

The interpretation of stress values can be greatly affected by the options used and by various parameters, such as the number of stimuli and the number of dimensions. While actual experience with real data has been the primary source of information so far, fortunately several papers have explored this question by Monte Carlo techniques. We recommend particularly the references by [Klahr 1969], [Sherman 1972], [Spence 1972], [Stenson & Knoll 1969], [Wagenaar & Padmos 1971], and [Young 1970].

It is possible to transform the data values by a fairly flexible formula, as a pre-processing step prior to the actual scaling computation. This can be a very useful preliminary step, both to help the main scaling procedure when linear or polynomial regression is being used, and to help the **`TORSCA`** initial configuration do a better job.

It is possible to weight the squared terms in **`STRESS`**. Let **`WW(1)`** to **`WW(MM)`** be a list of positive weights corresponding to the data values. The formula for stress now becomes:

$$STRESS = \sqrt{\frac{\sum_{m=1}^{MM} WW(m) * \left(DIST(m) - DHAT(m)\right)^2}{\sum_{m=1}^{MM} WW(m) * \left(DIST(m) - d_0\right)^2}}$$

KYST permits the weights to be given as a reasonably flexible function of the corresponding data values, or to be read in.

An important element of generality is the ability to "split" the data into separate sublists and perform separate regressions on each list. This "splitting" may be used to do unfolding. When the data is split, a separate regression and a stress is computed for each sublist by the formula above. The overall stress is taken to be the root-mean-square of the several sublist stresses. It is possible to split the data in several different ways. Also, it is possible to use different types of regression for different sublists, a fact which offers some interesting possibilities.

Normally, KYST uses a distance formula which is appropriate for Euclidean space. However, it is capable of using a certain other kind of distance which mathematicians usually call the L_p-metric, and occasionally the Minkowski *r*-metric. If points *x* and *y* have coordinates x_i and y_i, then the Minkowski *r*-metric distance between them is given by:

$$\text{distance}_{r(x,y)} = \left[\sum_i |x_i - y_i|^r \right]^{1/r}$$

For $r = 2.0$, this is ordinary Euclidean distance. For $r = 1.0$, this is the so-called city-block distance, or Manhattan metric. For $r = \infty$, it can be shown that this formula yields the maximum of the absolute coordinate differences, sometimes called the "dominance" metric which is a simple instance of the important distance mathematicians call the supremum or "sup" metric. For any value of r between and including 1.0 and ∞, a classical theorem states that the Minkowski r-metric is a true metric because it satisfies the triangle inequality. For values of r between but excluding 0 and 1.0, it is known that the Minkowski r-metric does not satisfy the triangle inequality although it does if one omits the outside exponent of $1/r$, but this case has nevertheless received much study. It appears that KYST should be capable of handling values of r in this range also, though we have no actual experience.

4. Output

The program always provides printed output. By use of the control phrase **CARDS**, it is possible to obtain punched card output.

The more important input cards are printed out, so it is easy to see what data and options were used. The final configuration obtained is also printed, and the corresponding value of the stress. (The stress is the "badness-of-fit quantity" which is minimized by the program.) For further explanation, see the references. Several other kinds of printed output are available on an optional basis. The phrase **PRINT=DATA** will cause a detailed listing of the input data. The phrase **PRINT=DISTANCES** will produce a detailed listing which includes the data, the distances, and the fitted regression values. The phrase **PRINT=HISTORY** will produce a "history" of the calculation which shows the progress of the iterative numerical calculation.

A printer plot of stress versus dimension is automatically provided, when appropriate. Two other types of graphical output are optional. How to obtain a printer plot of distances and fitted distance values versus data, as well as printer plots of the final configuration is explained in Section 12, Control Phrases for Output. These plots are all produced on the line printer, and require no special plotting equipment.

The only punched output provided optionally is a deck of cards called the "configuration deck". This deck contains the coordinates of the points of the final configuration, together with some preliminary cards, which permit it to be used as input to provide a starting configuration in a later scaling computation. For a description of the configuration deck, see page 19.

5. Options and Control Phrases

There are many options which can be chosen when using KYST. Despite the large array of possibilities, however, the user is able to specify the correct options without undue difficulty because of the system of free-format control phrases which is used. The simple "grammar" of control phrases is described in the next few paragraphs. The various options, and the control phrases which invoke them, are described in subsequent sections.

One important principle is used throughout. Whenever several alternative possibilities are available, one is always selected to be the "standard" or "default" choice. In the absence of any indication from the user, the program uses these standard choices. Thus the user is not burdened by having to specify all possible options. He need specify only those in which he departs from the standard choices.

Each line below contains one complete, legitimate control phrase:

```
DATA
DIAGONAL ABSENT
DIMMAX 4
DIMMAX=4
PLOT=CONFIGURATION=NONE
```

See Figure 4 at the end of the paper for other examples. In general, a control phrase may appear anywhere on a control card. It may consist of one or more words or numbers, separated by one or more spaces, equal signs, or commas.

The first item in a phrase must always be a word. Each item either terminates the phrase, or else determines a special set of allowable items which can follow it.

The program distinguishes sharply on control cards a "number-with-decimal-point" from an "integer". The key feature is the presence or absence of a decimal point. Where one kind of number is expected, the other is illegal.

Optionally, equal signs and commas ("=" and ",") may be used to improve the appearance of the control phrases. Thus a card may contain:

```
DIMMAX=5, DIMMIN=1
```

instead of the strictly equivalent:

```
DIMMAX 5 DIMMIN 1
```

The program which interprets the control phrases treats these two characters exactly like blanks. Collectively, these three equivalent characters are called break characters.

A single card may contain several control phrases. However each phrase must be complete on a single card. One or more break characters must separate distinct items (that is, words or numbers), and no break character may occur within a single item. It may be helpful to know that although control words may be much longer, only the first six characters are used and need be included.

The dollar sign may be used to enter a default parameter. This device is especially useful in the **DFUNCTION** and **WFUNCTION** options explained later, where five parameters must be entered. Thus the phrase:

```
WCONSTANTS = 2.0,$,$,$,-1.0
```

assigns default values to the second, third, and fourth parameters. It should be noted if **WCONSTANTS** is invoked more than once before a single **COMPUTE** card, a defaulted parameter will retain the value last assigned to it.

6. Basic Control Phrases

To initiate actual computation, use the control phrase:

COMPUTE

This must be on the very last card which specifies what you want done, as computation starts without examination of further input cards when the card containing this phrase has been read.

After the computation initiated by the **COMPUTE** card is complete, the program looks for further input. This makes it possible to perform several different and possibly unrelated computations on one computer run. Before taking further input, however, the program wipes the "slate clean" by initializing all parameters and options back to their normal values. A few very special exceptions to this rule are explained later.

Since the program always looks for further input after completing a computation, there must be some way of indicating that the entire computer run is to be ended. For this purpose, use the control phrase:

STOP

This must be on a card by itself.

7. Control Phrases for Data and Weights

This section describes the various ways KYST can interpret and handle the possibly weighted data. The actual reading in of the data is accomplished via a data deck, whose format will be described in detail in the next section. Since the control phrases mentioned below through **CUTOFF** specify how the data is to be handled when it is read in, they must appear on or before the control card containing the phrase:

DATA

which introduces the data deck.

In general, a single data deck may contain several groups of rows of data though in the simplest situations, a data deck will contain only one such group . Each group may be thought of as a square matrix, or part of a square matrix. Each group has the same size and shape as every other in the same data deck. Normally, the different groups serve as replicates, and contain different data relating to the same objects. However when the block diagonal option is in effect, the different groups are not replicates but pertain to different objects.

Each group in the data deck may have one of five forms:

MATRIX standard option
LOWERHALFMATRIX
UPPERHALFMATRIX
LOWERCORNERMATRIX
UPPERCORNERMATRIX

For the two halfmatrix forms, two variations are possible:

DIAGONAL=PRESENT standard option
DIAGONAL=ABSENT

MATRIX: In this case the group consists of **NPART** rows with each row expected to contain **NPART** matrix entries. The information presented pertains to **NPART** points or objects.

LOWERHALFMATRIX, **UPPERHALFMATRIX**: In these cases, the group consists of **NPART** rows if the main **DIAGONAL** is **PRESENT** and **NPART - 1** rows if the main **DIAGONAL** is **ABSENT** . Each row is expected to contain from 1 up to **NPART** or from 1 up to **NPART - 1** matrix entries in a regularly varying way. The information pertains to **NPART** points or objects.

LOWERCORNERMATRIX, **UPPERCORNERMATRIX**: In these cases, each group consists of **NROWS** rows, with each row expected to contain **NCOLS** matrix entries. The information pertains to **NCOLS + NROWS** points or objects. As the placement of the corner in the matrix would suggest, the first several points in the lowercorner case and the last several points in the uppercorner case correspond to the columns. Figure 2 on pages 17-18 , in which a data value corresponding to the (i,j) matrix entry is denoted by **ij**, should help make these forms clear.

The main use for the halfmatrix forms occurs when measured dissimilarity or similarity between objects i and j is experimentally symmetric, so there is no conceptual distinction between the (i,j) entry and the (j,i) entry. One major use for the cornermatrix forms is in unfolding. Because the "splitting" option see below is restricted to rows and not available for columns, the columns must correspond to stimuli or "real" points and the rows to subjects (or "ideal" points).

When a data deck contains more than one group, normally the different groups pertain to the same objects, and contain replicated measurements of the same matrix entries. Thus if a single group pertains to **NPART** objects, then normally the whole data deck pertains only to **N = NPART** objects. However, when the blockdiagonal option is used, each group pertains to a new set of objects, and the several groups should be thought of as blocks down the diagonal of a single larger data matrix. If each group pertains to **NPART** objects, and there are **NREPL3** groups, then the whole data deck pertains to **N = NREPL3 * NPART** objects. If each group is only part of a square matrix, it is the square matrices which are strung down the diagonal, with each group occupying its appropriate position in its square matrix. The shape of the groups is determined by using the above control phrases to describe the data. The block diagonal option is controlled by the phrases:

BLOCKDIAGONAL=NO standard option
BLOCKDIAGONAL=YES

Two possible configurations using the **BLOCKDIAGONAL=YES** option are the following:

(?)

As always, the standard option applies if no specific indication is given.

If individual data values are missing, this fact may be indicated by using values which are algebraically less than some specific cutoff value, and indicating this value by the phrase:

CUTOFF=number-with-decimal-point (standard value is -1.23*10E20) .

Values which are < **CUTOFF** value will be discarded. Since this discarding takes place while the data is being read in, the **CUTOFF** must be provided on or before the **DATA** card. The standard value is so large a negative number that it is unlikely to affect anyone's data.

In addition to all the options mentioned above, it is possible to have not one but several data values for each matrix entry of each group. This is called internal replication. While this feature is handled by the parameter card of the data deck rather than by a control phrase, it is mentioned here merely for clarity.

Note again that if used, all the phrases above must appear before the data deck. Also note that if more than one data deck is provided, only the first one is used by the **TORSCA** initial configuration routine, though all are of course used in the main scaling routine.

Following the first computation initiated by a **COMPUTE** card , it is possible to use the data from the preceding computation without another copy of the data deck. It is simply necessary, somewhere before the relevant **COMPUTE** card, to use the phrase:

SAVE = DATA

The effect of this phrase is simply to retain for this computation the data from the last computation. It is even possible to use **SAVE=DATA** and follow it by another data deck. This will have the same effect as using more than one data deck in one computation, which is explained elsewhere.

The use of **SAVE=DATA** requires some care. All items which the program stores internally with the data are frozen by use of this option, and it may not be obvious just which items these are. Everything which must be specified before the data deck is stored with the data, and will be retained by use of a **SAVE=DATA** card, regardless of other option cards accompanying the **SAVE=DATA** card. Such frozen options include regression-type and shape of matrix.

Actually, data is saved without use of a **SAVE=DATA** card, if no new data is provided. To add a new data deck without losing the old data from before the last **COMPUTE** card, however, it is necessary to **SAVE=DATA** before the new data deck.

When performing metric scaling, a preliminary transformation of the data may be quite important. Also, regardless whether the scaling is metric or not, such a preliminary transformation may be helpful to the classical Torgerson scaling which is the first step in the **TORSCA** initial configuration option, by making the data values more like Euclidean distances. Any transformation desired can of course be accomplished by a simple computer program specially written for the purpose, and the transformed data then entered into KYST; also a still simpler function subroutine **DTRAN** may be used with KYST as described below. However, to encourage the use of transformations we have included the following transformations directly within KYST:

$$DATA = s + t * (a + b * DATA)^p$$

In words, this transformation consists of a first stage linear transformation using a and b, a second power stage transformation using p, followed by a third stage linear transformation using s and t. To invoke this transformation, place the following two cards immediately after the data deck (except that weight generating cards to be described below are permitted to intervene):

DCONSTANTS = a,b,p,s,t

where each of the five parameters
is a number-with-decimal-point
(standard values 0.0, 1.0, 1.0, 0.0, 1.0);

DFUNCTION

While the **DCONSTANTS = a,b,p,s,t** phrase must precede the **DFUNCTION** phrase to be effective, it could occur anywhere before **DFUNCTION** and need not be placed as shown. Recall that a dollar sign may be used instead of a number to indicate the default value. KYST uses a function subroutine called **DTRAN** to accomplish the transformation. If the user provides his own FORTRAN function **DTRAN(X)**, **DFUNCTION** will invoke its use.

Weights associated with the data values are sometimes useful, to reflect the varying measurement errors or for other reasons. How they are used by KYST is indicated in an earlier section. Weights may either be computed as a function of the associated data values, or read in from a deck. If no weights are provided by either method, the program supplies the default weight of 1.0 for every data value. The weights are effective both during the initial configuration calculation, and during the main scaling computation. It is dangerous to use weights of 0.0, since they may lead to division by zero in the monotone regression fitting algorithm **FITM**. Negative weights have not been considered. The effect of a zero weight can be obtained by causing the corresponding data value to be missing see **CUTOFF** above, or approximated by using a very small positive weight.

It is possible, directly within KYST, to compute the weights as any function of the data of this type:

$$WEIGHT = s + t * (a + b * DATA)^p$$

In words, this transformation consists of three stages, a first linear stage using a and b, a second power stage using p, and a third linear stage using s and t. To invoke this transformation, place the following two cards immediately following the data deck or immediately following the **DFUNCTION** card:

WCONSTANTS = a,b,p,s,t

where each of the five parameters
is a number-with-decimal point
(standard values 0.0, 1.0, 1.0, 0.0, 1.0);

WFUNCTION

If both **WFUNCTION** and **DFUNCTION** are used, it is vital to place them on separate cards, and important to remember that the corresponding operations are invoked in the same order in which the cards appear. Thus if **DFUNCTION** appears first, the weights will be computed from the transformed data values. While the **WCONSTANTS = a,b,p,s,t** phrase must precede the **WFUNCTION** phrase to be effective, it could occur anywhere before **WFUNCTION** and need not be placed as shown. Recall the possible use of for default values. KYST uses a function subroutine **WTRAN(X)** to accomplish the transformation. The user may supply his own version, and invoke its use.

It is also possible to read in weights from a deck. To introduce the deck the control phrase:

WEIGHTS

is used. When a weights deck is used, it must immediately follow the data deck, with no intervening cards except possibly for **DCONSTANTS** and **DFUNCTION** . Generally speaking, the arrangement and format of the weights deck must be exactly the same as that of the data deck, and one weight must appear for every data value, even those which are artificial and indicate missing data. For details, see the section on deck formats.

8. Data and Weights Decks

There are three types of input decks: data decks, weights decks, and initial configuration decks. Only the data deck is necessary.

Discussion of the initial configuration deck will be deferred to the next section.

The data deck consists of both non-control cards and one control card which introduces it. Except for the initial card, the cards of the data deck are not control cards. Similarly, each of the other kinds of decks consists of one initial card which introduces it, and subsequent cards which are not control cards.

The first card of a data deck is a control card containing **DATA**. Naturally other control phrases may appear on this card also. It is often convenient to include on this card phrases such as **LOWERCORNERMATRIX**, **UPPERCORNERMATRIX**, and **CUTOFF = 0.5** which are likely to be permanently attached to this data. The second card is a title card, which acts as a label for the data. The title card may contain any text at all. The third card of the data deck is a parameter card. It should contain either three or four integer parameters, placed so that they end in columns 3, 6, 9, and l2.

Before describing the arrangement of parameters, let us recall that in general the data deck consists of several "groups" (see section 7) of rows. Each group has a certain number of rows and columns the same for every group in a single data deck . Unless one of the cornermatrix options is in effect, the number of rows is the same as the number of columns. In addition to the replication possible by using several groups, internal replication is possible: each matrix entry from a single group may be represented by several adjacent data values. The number of them is constant throughout the data deck, and is called the number of replications.

Unless one of the cornermatrix options is in effect, the parameter card should contain three parameters, as follows:

1) the number of rows and columns in each group (**NPART**);

2) the number of replications (**NREPL1**);

3) the number of groups (**NREPL3**) .

If one of the cornermatrix options is in effect, the parameter card should contain four parameters, as follows:

1) the number of rows in each group (**NROWS**);

2) the number of columns in each group (**NCOLS**);

3) the number of replications (**NREPL1**);

4) the number of groups (**NREPL3**).

The fourth card of the data deck should contain a format, constructed according to the rules for FORTRAN format statements. For computers which distinguish between floating-point numbers and integers internally (and many do) , it is necessary to use format conversion characters like E and F which correspond to floating-point numbers. The format should be appropriate for a single row of a single group of the data deck, since KYST uses a single FORTRAN READ statement with this format to read a single row. If you need information about FORTRAN formats or READ statements, consult a FORTRAN manual or a programmer.

Following the format card are the data values themselves. A new row should always start on a new card. Aside from this, these cards may be punched in any way you desire, consonant with the format you have provided. Where a row is too long to fit on a single card, it may be placed on as many cards as necessary.

More than one data deck may be used. When this is done, the different decks are related to one another more or less as the different groups within a single data deck. The different data decks pertain to the same set of objects as one another. However different data decks may use different types of regression, and may have different shapes, different numbers of replications, and different formats. Only the first data deck is used by the **TORSCA** initial configuration routine, though all are of course used during the main scaling.

The first card of a weights deck is a control card containing **WEIGHTS**. The subsequent cards contain the weights themselves, which should not be negative nor zero. The weights decks is required immediately to follow the corresponding data deck without intervening cards except possibly **DCONSTANTS** and **DFUNCTION** cards . KYST reads the weights in the same way and with the same format it used for reading the preceding data. Thus the cards containing the weights should be identical in arrangement and format to the cards containing the data values. Where a data value is indicated as being missing because it falls below the value of **CUTOFF**, a corresponding value must nevertheless appear in the weights deck, even though it will not be used. A blank value can generally be used in this case. Aside from weights corresponding to missing data values, weight values of zero are dangerous, as explained elsewhere.

When more than one data deck is used to supply data for a single computation, every data deck should be given the same weights treatment. Either no data deck should be weighted, or every data deck should be accompanied by a weights deck, or every data deck should use computed weights. The weights decks, **WCONSTANTS** and/or **WFUNCTION** cards should be interlaced with the data decks.

Examples of Data Decks

Entry for (i,j) position in matrix has value **ij**. Actual data should consist of floating-point numbers.

```
DATA=MATRIX
title card
4 1 1
format card
11 12 13 14
21 22 23 24
31 32 33 34
41 42 43 44

DIAGONAL=PRESENT
DATA,LOWERHALFMATRIX
title card
4 1 1
format card
11
21 22
31 32 33
41 42 43 44
```

```
DIAGONAL=ABSENT
DATA,LOWERHALFMATRIX
title card
4 1 1
format card
21
31 32
41 42 43

DIAGONAL=PRESENT
DATA,UPPERHALFMATRIX
title card
4 1 1
format card
11 12 13 14
22 23 24
33 34
44

DIAGONAL=ABSENT
DATA,UPPERHALFMATRIX
title card
4 1 1
format card
12 13 14
23 24
34

DATA,LOWERCORNERMATRIX
title card
3 2 1 1
format card
31 32
41 42
51 52

DATA,MATRIX
title card
2 3 2
format card
11 11 11 12 12 12
21 21 21 22 22 22
11 11 11 12 12 12
21 21 21 22 22 22

BLOCKDIAGONAL=YES
DATA,MATRIX
title card
2 1 3
format card
11 12
21 22
33 34
43 44
55 56
65 66
```

Figure 2

9. Control Phrases and Deck for Initial Configuration

There are five ways in which the initial configuration may be produced:

1) the **TORSCA** initial configuration procedure may be used;

2) a random initial configuration may be used;

3) the so-called "**ARBITRARY**" initial configuration may be used;

4) a configuration may be read in from a deck;

5) the solution from a preceding computation may be saved for use as the starting configuration.

Note that when a single **COMPUTE** card is used to initiate scaling in several different dimensionalities, the choice above applies only to the first scaling, which is in the highest dimensionality. For each subsequent scaling the solution from the last previous scaling in a higher dimensionality is used with the last one or more coordinates dropped off. If rotation to principal coordinates is being used under the standard option **COORDINATES = ROTATE**, this procedure usually provides a very good starting configuration.

We list here the five corresponding control phrases:

TORSCA standard option;
RANDOM=integer (standard value for integer is 0)
ARBITRARY
CONFIGURATION
SAVE=CONFIGURATION

Further explanation follows.

Under the control phrase **TORSCA**, KYST first uses the classical Torgerson scaling technique to derive a configuration. If the number of points N < 60, the program then proceeds to use the quasi-nonmetric Young method of improving this configuration. The control phrase:

PRE-ITERATIONS=integer (standard value is 1)

determines the maximum number of such steps. If N > 60, however, memory space does not permit the Young procedure to be used. If more than one data deck is used, as described elsewhere, the **TORSCA** initial configuration calculation will use only the data from the first deck. In some cases, the initial configuration obtained from the **TORSCA** procedure is already quite good.

If **RANDOM**=integer is used, the program generates its own starting configuration by picking pseudo-random points from an approximately normal multivariate distribution. Actually, each coordinate is obtained by applying the logistic transformation to a uniform random variable, obtained from a moderate-quality portable generator. This feature is especially valuable when you wish to generate many solutions starting from different configurations, in order to guard against local minimum solutions. Use of **RANDOM**=integer in each of a successive set of computations during a single computer run will produce new and independent starting configurations. However, the pseudo-random number generator naturally produces the same sequence of numbers in each computer run. In order to permit different starting configurations in different runs, the program obtains and discards several random numbers before generating the initial configuration. The integer which follows **RANDOM** specifies how many random numbers should be discarded. By altering this integer from one run to the next, you can make the program use different starting configurations on different runs.

The control phrases **ARBITRARY** is included primarily for historical compatibility. If it is used, the starting configuration prior to being standardized is given by:

$$\text{r - th coordinate of the i - th point} = \begin{cases} 0.01 & \text{if r = (i mod dimensionality of the i - th point)} \\ 0 & \text{otherwise} \end{cases}$$

The control phrase:

CONFIGURATION

introduces a deck containing a configuration. The subsequent cards of this deck are not control cards, and must have the following precise arrangement which parallels that of the data deck, described in the previous section. The second card of the configuration deck, which immediately follows the **CONFIGURATION** control card, is a title card, which serves as a label for the configuration. It may contain any suitable text, and is printed out where appropriate. The third card is a parameter card. It should contain two integer parameters, placed so that they end in columns 3 and 6. The first parameter indicates the number of points in the configuration. The second parameter indicates the number of dimensions, that is the number of coordinates in each point. The fourth card is a format card, containing a FORTRAN format statement suitable for reading all the coordinates of a single point. The subsequent cards of the configuration deck contain the configuration itself. These must be arranged point by point, with all coordinates for a single point together. The coordinates for each point must begin on a new card, though as many cards as are needed may be used for each point.

Following the first entire computation that is, all the scalings initiated by a single **COMPUTE** card, use of the phrase:

SAVE=CONFIGURATION

will retain as the starting configuration for the next computation the last solution from the preceding computation. This is sensible only if the dimensionality of the last solution is at least as great as the dimensionality of the new computation.

Using either of the control phrases **CONFIGURATION** or **SAVE=CONFIGURATION** it is possible to provide a configuration which does not have enough points to serve as an entire starting configuration. If no points are provided, one may consider that an initial configuration with zero points has been provided. By default enough additional points are provided arbitrarily. The phrase **RANDOM**=integer can be used to provide the remaining points randomly to complete the initial configuration. Thus it is possible to use the phrase **SAVE=CONFIGURATION** or a configuration deck, followed by either **ARBITRARY** or **RANDOM**=integer.

10. Control Phrases to Choose Analysis Type

A single **COMPUTE** card can initiate several computations, all using the same data and same options, in several spaces of different dimension. The first computation is done in the maximum dimension **DIMMAX**. Then the dimension drops by the dimension difference **DIMDIF**, and another computation is done. This is repeated until finally the computation is done in the minimum dimension **DIMMIN**, which completes the computation caused by the present **COMPUTE** card. The phrases:

DIMMAX=integer (standard value is 2);
DIMMIN=integer (standard value is 2);
DIMDIF=integer (standard value is 1);

control the dimensions actually used.

The kind of distance function used by KYST is controlled by the phrase:

R=number-with-decimal-point (standard value is 2.0).

For **R**=2.0, this yields the ordinary Euclidean distance, while for **R**=1.0, this yields the city-block metric. For these two special values, the necessary powers and roots are computed in a relatively rapid way, but for other values the general purpose power operations are used, which behind the scenes require a logarithm and an exponential to be calculated for each power operation. Thus for general values of **R**, the computation will be perceptibly slower.

Two different stress formulas are available, as indicated earlier. The phrases:

SFORM1 standard option

SFORM2

choose the one to be used. As explained in detail above, formula 2 uses a variance-like expression involving **DBAR** in the denominator, while formula 1 uses 0 in place of **DBAR**.

As explained in detail in [Kruskal l964a and l964b], there are two ways of treating ties among the data values when performing a monotone regression. Consider the fitted regression values corresponding to a group of equal data values. Either no restriction is placed on the size of these fitted values with respect to one another the "primary" approach , or these fitted values may be required to be equal the "secondary" approach. The following control phrases:

PRIMARY standard option

SECONDARY

control this option. Where there are few ties, it hardly makes any difference which is used. Where there are a great many ties, as when the data values represent coarse category judgements, it does make a difference, and the context must be considered in making the choice.

The regression type may be either monotone or polynomial. The former makes the scaling nonmetric, the latter metric. The monotone regression may be either ascending suitable for dissimilarities or descending suitable for similarities. The polynomial regression may be of any degree up to quartic degree 4 , with or without constant term. The coefficient values are a by-product of the regression. These phrases:

REGRESSION=ASCENDING standard option
REGRESSION=DESCENDING
REGRESSION=POLYNOMAL=integer degree of polynomial
REGRESSION=CONSTANT standard option
REGRESSION=NOCONSTANT
REGRESSION=MULTIVARIATE=integer number of variates

which control regression type, must be given before the data deck. The integer in **REGRESSION=POLYNOMIAL**=integer gives the degree of the polynomial, which should be no more than four. It is possible to include or exclude the constant term when fitting polynomials by use of **REGRESSION=CONSTANT** and **REGRESSION=NOCONSTANT**. The last phrase, **REGRESSION=MULTIVARIATE=** integer, is needed only in special cases, as indicated below.

Regression type must be indicated before the data deck or on the card which introduces the data deck , because it is stored away with the data. When more than one data deck is used, a different type of regression may be chosen for each data deck, if desired. The former restriction helps permit the latter freedom. There are situations where it may be helpful to use monotone and straight-line regression together thus performing something intermediate between nonmetric and metric scaling .

Some useful types of regression may be quite easily substituted for polynomial regression, by writing a short function subroutine to replace the very simple program **REGR**. If **REGR(X,K)** provides the function value $f_k(x)$, then KYST will perform the standard multivariate linear regression procedure and pick out the best linear combination of the functions f_k. The phrase **REGRESSION=MULTIVARIATE**=integer should be used to indicate how many functions are included. Because a very simple method is used to solve the simultaneous linear equations, it is desirable that the functions f_k should not be too terribly far from orthogonal. Not more than five functions f_k are permitted.

The entire list of data values may be split into sublists, with a separate regression performed on each sublist. There are several ways in which the data can be split. Each row of every data deck can be a sublist, each group of rows (see section 8) can be a sublist, each data deck can be a sublist. Where there is more than one data decks, different decks may be split differently. These phrases:

SPLIT=BYROWS
SPLIT=BYGROUPS
SPLIT=BYDECKS standard option
SPLIT=NOMORE

which control the splitting, must occur before the data decks to which they pertain, because they affect how the data is stored away. **SPLIT=BYROWS**, **SPLIT=BYGROUPS**, or **SPLIT=BYDECKS**, make each row, each group of rows, or each data deck a separate sublist respectively. **SPLIT=NOMORE** is relevant only when several data decks are used. It causes all subsequent data decks to be joined into a single sublist, until another **SPLIT** card. If it is given before the first data deck and no more splitting instructions occur, there will be only one list. There is an additional feature which is useful when there are several data decks, and you wish to join them into two or three sublists. Even though splitting has been totally suspended by an earlier use of **SPLIT=NOMORE**, a subsequent use of **SPLIT=NOMORE** will cause a split where it occurs, though splitting will continue suspended. KYST can handle up to l00 sublists.

The phrase:

FIX=integer (standard value is 0),

fixes the first several points of the configuration, so that they do not move during the main scaling computation. In other words, the program treats these points as given, and not subject to adjustment. This option is sensible only if the points of the initial configuration which have been fixed are sensible. Thus normally this option would be used only if points are read in from a configuration deck, or if the initial configuration has been created by the **TORSCA** initial configuration routine. Note that the **FIX** option has no effect on how the **TORSCA** initial configuration routine operates, but only affects the main scaling procedure. The integer indicates how many points are fixed.

If the integer is 0, then the phrase has no effect. Only the initial points of the configuration can be fixed, that is, the points numbered from 1 up to the value of the integer. The means used internally to accomplish this fixing is to set to zero all the coordinates of the gradient vector which correspond to fixed points, just before the program starts to make use of the gradient vector. If the user wants these values to appear without change as the coordinate values in the final configuration, he should use the phrase **COORDINATES=AS-IS.** Coordinate options are explained fully later. Otherwise the fixed points, along with the rest of the configuration will be subject to the usual standardization and rotation, and the resultant coordinate values will be different than those fixed.

Note that the number of points in the configuration deck is independent of the number of points fixed by this new phrase, and that both of these are independent of the number N of stimuli in the data deck. Thus it would be possible and perhaps even sensible to use a data deck which pertains to 30 stimuli that is, points, read in a starting configuration which pertains to only 10 stimuli, and **FIX** only the first 7 of these points.

11. Phrases to Control Termination and Convergence

Normally computation runs through enough iterations, stopping when the program judges that it has reached the minimum stress value. To control computation time in case convergence is unduly slow, however, the program will terminate computation after 50 iterations and yield normal output even if a minimum has not yet been reached. The reason for termination is always indicated. To change the maximum number of iterations, use:

ITERATIONS=integer (standard value is 50) .

More iterations say l00 may be useful to assure complete convergence, though the further change after 50 iterations if often so small as to be negligible. Fewer iterations say 25 or less may be useful where a great many sets of data are to be analyzed and computer time is to be conserved.

Of course, the program has no way of distinguishing between a local minimum and a global minimum. That is, any minimum it reaches may be merely the smallest value in some local region a local minimum rather than the smallest value everywhere a global minimum . This difficulty is shared by all procedures for minimizing functions of many variables except when very special conditions hold, such as convexity. For the more usual ways of using KYST, the **TORSCA** initial configuration option should reduce the chance of reaching a merely local minimum. For further information about this question, consult the growing literature, including papers such as [Spence l972] . If you suspect that a solution may not be the global minimum, one standard technique is to repeat the scaling several times, starting from different initial configurations as explained in [Kruskal l964b]. The use of **RANDOM**=integer and **SAVE=DATA** makes this easy to do. As explained in the reference, if the same solution up to such immaterial changes as rotation and reflection occurs several times and has the least stress by a considerable amount of all solutions found, while only a few other solutions occur, which are all worse and all quite different from one another, then we may have considerable practical confidence in having found the true minimum solution.

At a minimum configuration whether global or merely local the gradient vector is 0 that is, all the partial derivatives of the stress are 0. One test the program uses for determining whether it has reached a local minimum is whether the length of the gradient is small enough. The phrase:

SFGRMN=number-with-decimal-point (standard value is 0.0).

provides a criterion value. **SFGRMN** stands for scale factor of the gradient minimum. When the length more properly, the scale factor of the gradient is less than or equal to this criterion, the program decides that a local minimum has been reached. Due to the fact that computer arithmetic is approximate, it is very unusual for the gradient to become precisely 0. Thus with the standard value in effect, this test is seldom satisfied.

In practice, primary reliance is placed on how fast the stress is decreasing. If the ratio between present and previous stress is close enough to 1.0, and if simultaneously the weighted average of past values of this ratio is close enough to 1.0, then the program decides that a local minimum has been reached. The control phrase:

SRATST= number-with-decimal-point (standard value is 0.999)

provides the criterion value. **SRATST** means stress ratio stop. If both the stress ratio and the weighted average Stress ratio are between this value and l.0, then the criterion is met. A more rigorous criterion is achieved by using a value like 0.9999 which is closer to l.0, while a more generous criterion comes from a value like 0.99 further from 1.0. If 1.0 itself is used, the criterion becomes virtually unachievable.

Even if a minimum value of stress has not yet been achieved, it is sometimes desirable to stop computing when the stress has become sufficiently small. The control phrase:

STRMIN=number-with-decimal-point (standard value is 0.01)

provides the criterion value. **STRMIN** means stress minimum. Usually when stress gets smaller than this (that is, below 0.01), only negligible further movement of the configuration occurs in achieving a minimum. For nonmetric scaling, if the stress becomes small enough, in practice it turns out that zero stress is always achievable. However, it may take many iterations of almost infinitesimal movement to achieve it.

During computation, the cosine of the angle between successive gradients plays an important role in several ways. The exponentially weighted average of past cosine values is kept up to date, and a similar average of their absolute values. The control phrases:

COSAVW=number-with-decimal-point (standard value is 0.66);
ACSAVW=number-with-decimal-point (standard value is 0.66);

may be used to alter the weights used in forming these averages.

Whenever computation is terminated, the reason is always indicated by a remark such as **MINIMUM WAS ACHIEVED, MAXIMUM NUMBER OF ITERATIONS WERE USED, SATISFACTORY STRESS WAS REACHED,** or **ZERO STRESS WAS REACHED**

12. Control Phrases for Output

Normally no punched card output of any kind is produced. Sometimes however it is useful to get the final configuration out on a configuration deck, either for use as a starting configuration in further scaling, or for other analysis. These two phrases control the punchout option:

NOCARDS standard option
CARDS

There are three kinds of optional printed output in addition to the basic printed output which is always supplied. The phrases:

PRINT=NODATA standard option
PRINT=DATA

control whether the data and weights are printed out at the beginning of the computation. This printout is sometimes useful for reference, or if KYST seems to misunderstand your input data. The phrases:

PRINT=HISTORY standard option
PRINT=NOHISTORY

control whether the program prints certain information on each operation. This "historical" record of the computation is useful in seeing how most of the stress reduction takes place at first, and also permits someone experienced in the art to judge such things as how close to the minimum configuration the program has actually gotten. The phrases

PRINT=NODISTANCES standard option
PRINT=DISTANCES

control whether a rather voluminous terminal printout takes place. This printout shows the data values in order of size in each sublist, if splitting has been used , and with each data value shows the indices of the two points it pertains to, the distance between the two points, and the corresponding regression value.

There are also three kinds of graphical output produced on the printer. If the data is analyzed in more than one dimension, a plot of stress versus dimension is produced. The two other types of plots are optional, governed by control phrases. The phrases

PLOT=SCATTER=NONE
PLOT=SCATTER=SOME standard option
PLOT=SCATTER=ALL

control whether a scatter plot of the distances **DIST** and the fitted distances **DHAT** versus the data **DATA** is generated. If the option **NONE** is used, no scatter plotting is done. If the data is split into more than one sublist, then will give a separate plot for each sublist, as well as a plot for the entire data. **SOME** is similar, but plots only the first five sublists. If the data is not split, **SOME** and **ALL** both have the same effect: a single plot of distances and regression values versus data values.

The phrases

PLOT=CONFIGURATION=NONE
PLOT=CONFIGURATION=SOME
PLOT=CONFIGURATION=ALL standard option

control plotting of the final configuration. The use of **ALL** produces $LDIM*(LDIM-1)/2$ plots of all pairs of dimensions of the configuration, where **LDIM** is the dimensionality of the configuration. The option **SOME** is similar except that each dimension is plotted against only one other dimension. However, if **LDIM** is odd, dimension l is plotted twice, once with dimension 2, and once with dimension 3. If **NONE** is used, no configuration plotting is done.

Several phrases pertain to the form of the configuration:

COORDINATES=AS-IS
COORDINATES=STANDARDIZE
COORDINATES=ROTATE standard option

Under the **STANDARDIZE** option, the configuration has its centroid set to the origin and the mean square distance of the points from the centroid set to 1 after every iteration. The option **ROTATE**, beside causing the above standardization on each iteration, also causes the final configuration to be rotated so that its principal components lie along the coordinate axes. Where the scaling is to be done in several different dimensionalities, this has the desirable effect of causing the least important coordinates to be dropped when going to the next lower dimensionality. The option **AS-IS** suppresses both the standardization and the rotation, and may be particularly useful when the initial configuration is read in from a configuration deck, particularly if the **FIX** option is used.

13. Some Ways to Use KYST

For nonmetric scaling of the most familiar type, the program may be used as in the preliminary example, with the following simple changes used where appropriate. The phrases **TORSCA** and **COORDINATES=ROTATE** are standard options and may be omitted. The **PRE-ITERATIONS** phrase will generally be omitted, since the standard option of l pre-iteration usually suffices. If the data are dissimilarities, the phrase **REGRESSION=ASCENDING** should be substituted for the corresponding phrase in the example. If the data form a lower or upperhalfmatrix, the phrases **LOWERHALFMATRIX** and **UPPERHALFMATRIX** should be used, with **DIAGONAL=ABSENT** if the diagonal values are missing. For metric scaling, the control phrase **REGRESSION=POLYNOMIAL**=degree should be included before the data deck accompanied by **REGRESSION=NOCONSTANT** if desired. Also, consider the possibility of transforming the data preliminary to the scaling, by the use of **DCONSTANTS=a,b,p,s,t** and **DFUNCTION**. If the data are transformed in connection with nonmetric scaling to improve the operation of the **TORSCA** initial configuration routine , note that the choice of the regression as **ASCENDING** or **DESCENDING** should be based on the form of the data after transformation.

To do unfolding, the rankings provided by each subject or judge should form one row of the data matrix, and the stimuli or objects judged should correspond to the columns. It is necessary to use either **LOWERCORNERMATRIX** or **UPPERCORNERMATRIX** before the data.

If larger distances from the subjects ideal point should match larger data values, the phrase **REGRESSION=ASCENDING** should be used, while if a reverse relationship between distances and data values is appropriate, the phrase **REGRESSION=DESCENDING** should be used. It is also possible to use KYST for nonmetric unfolding in which the subjects have negative ideal points instead of positive ideal points, that is, most disliked central points instead of most liked central points. In fact, the program has no knowledge of which kind of ideal point you are postulating, nor does it care. The only important fact for the program is whether distances from the ideal point, regardless of type, should have an ascending or descending relationship to the data.

When an unfolding is to be done nonmetrically, it is in theory vital to **SPLIT=BYROWS** and use **SFORM2**, as otherwise a meaningless zero-stress solution may be possible. This solution is one example of what is referred to as degeneracy.

In practice, however, the situation is not so clear, particularly if a good initial configuration is used. If the unfolding is metric, most often it will be preferable to **SPLIT=BYROWS**, but where rankings are reasonably comparable among subjects and the data is scanty, it can be sensible to include the rankings of all subjects in a single regression.

Suppose several subjects have provided direct judged dissimilarities of a single set of objects. Suppose either a) that the values provided by each subject form a separate data deck, or alternatively b) that there is only one data deck and that the values provided by each subject form a group of rows in the deck. There are at least four meaningful levels of analysis possible, based on two independent variables:

C1: A single configuration is shared by all subjects;
CS: A separate configuration is formed for each subject;
R1: A single regression is shared for all subjects;
RS: A separate regression is done for each subject.

Case C1R1 does not allow for any differences between subjects at all, and treats the different subjects merely as replications. Each of C1RS and CSR1 allows for differences between subjects; though in quite distinct ways. CSRS is conceptually and actually equivalent to performing an entirely separate scaling for each subject.

	R1	RS
C1	**a SPLIT=NOMORE** **b SPLIT=BYDECKS** **BLOCKDIAGONAL=NO**	**a SPLIT=BYDECKS** **b SPLIT=BYGROUPS** **BLOCKDIAGONAL=NO**
CS	**a Not possible** **b SPLIT=BYDECKS** **BLOCKDIAGONAL=YES**	**a Not possible** **b SPLIT=BYGROUPS** **BLOCKDIAGONAL=YES**

This table shows how to accomplish the various possibilities. Note that these four cases correspond to McGee's four cells [McGee, 1968]. However he includes a kind of distance between the separate configurations in his badness-of-fit measure, which gives his cases corresponding to our CS cases a somewhat different meaning than ours.

In an unfolding procedure using only linear (or polynomial) regression, KYST places no restriction on the coefficients of the regression function. Thus over the region containing the data values the regression functions can be ascending, descending, constant, or (in the case of polynomials of degree ≥ 2) ascending in part and descending in part. One way to handle situations where this freedom is undesirable is discussed below. One useful possibility that comes with this freedom is that a subject's "ideal" point may be a negative ideal, his most disliked possible stimulus, just as easily as a positive ideal, and the recovered shape of the regression line indicates which it is. One unfortunate concomitant is that where a subject's ideal point (positive or negative) truly belongs outside or on the edge of the region containing the real stimulus points, it is easy for the program to misplace it on the diametrically opposite position and make it of opposite type.

Sometimes it is desirable to do a scaling or an unfolding using linear (or polynomial) regression, but it is necessary to assure that the regression function is essentially monotone over the region containing the data values. While KYST cannot manage quite this, it can approximate it. The trick is to feed in the selfsame data twice, that is, use duplicate copies of the data deck. In front of one copy the phrase **REGRESSION=POLYNOMIAL**=integer is used. In front of the other copy the phrase **REGRESSION=ASCENDING** or **REGRESSION=DESCENDING** is used. Thus the overall stress reflects both deviation from the desired polynomial fit and deviation from the appropriate monotonicity.

14. User Limitations and Computing Times

As presently written, KYST has just a few primary limitations on the size of the problems it can handle. Up to 100 objects (that is, points) may be used. Scaling can be done up to six dimensions (and down to one dimension). Up to 1800 data values may be used. In this regard, it is not necessary to count data values which are formally present, but discarded because they are below the `CUTOFF` value. (However, note that each row is read in completely, before the data values in it are discarded. Thus if the number of data values including those below the cutoff exceeds 1800, it is possible though unlikely that the available storage space will be overrun and trouble result. This can happen only if the data values below `CUTOFF` are concentrated near the end of the data deck, so that at some point the number of genuine data values in prior rows, plus the total number of data values in the current row, exceed 1800.) If the data are split into separate sublists, up to 100 sublists can be accommodated. For metric scaling, the regression can be linear, or polynomial of up to fourth degree.

Computer running time will of course vary widely with the data, the options chosen, and the computer. However, we believe that the running time associated with a single `COMPUTE` card should be roughly proportional to (number of data values) times (total number of iterations, summed over all dimensionalities)

For one reasonable choice of options, the constant of proportionality on the Honeywell 6000 series is approximately three milliseconds. It must be noted, however, that this is a very rough estimate, and includes overhead CPU time which is not proportional to the length of the computation.

15. Some Precautions: Number of Dimensions, Convergence, and Local Minima

If there are not very many objects, it is impossible to recover many dimensions (regardless of whether many dimensions are in fact operative). This is because the number of parameters estimated needs to be considerably less than the number of observations (if the observations are subject to statistical fluctuation), a general principle of wide application. Blind use of the program can easily extract more dimensions than could possibly be meaningful.

Valuable information on the accuracy of recovery may be found in the papers by [Young 1970] and [Sherman 1972], pertaining to the most common type of scaling, namely, nonmetric scaling based on a halfmatrix without diagonal and only a single replication. While it is difficult to give hard and fast rules, the papers above and practical experience suggest the bare minimum number of objects for wise use under the circumstances quoted:

for a 1-dimensional solution, 5 objects;
for a 2-dimensional solution, 9 objects;
for a 3-dimensional solution, 13 objects.

Furthermore, if the number of objects is relatively small, each additional object will increase the accuracy of reconstruction quite a bit.

KYST finds the configuration of minimum stress by using an iterative procedure (of the "steepest descent" type) which starts with an initial configuration and keeps modifying it until, hopefully, it converges to the configuration having minimum stress. Two questions should always be asked about the solution yielded by such a procedure: (1) Is convergence complete? In other words, did the procedure finish converging to a minimum, or did it stop halfway? If convergence is not complete, you may not want to even look at the solution, but rather put it back in for further iteration. (2) Was the right minimum reached? In other words, did the procedure converge to a "local minimum" configuration, from which no small modification can decrease the stress, but at which the stress is greater than the true minimum value? If you fear the solution is only a local minimum, you may want to perform further scalings using other starting configurations to check this possibility.

Strictly speaking, convergence to an exact minimum is only complete if the partial derivatives of the stress are all exactly equal to 0. Because a computer does not represent numbers with perfect precision, this does not happen very often. However, in most practical situations it is possible to come so very close to the minimum towards which the procedure is converging that the difference can have no practical importance. In the vast majority of cases, moreover, other considerations (such as the statistical fluctuations in the data, or the fact that the model is not perfectly appropriate) mean that the achievable precision is far greater than necessary.

Thus the practical question is whether convergence has proceeded far enough to be practically complete. The program uses several tests to determine this (such as how fast the stress has been decreasing from iteration to iteration, and the size of the gradient), and declares that a minimum has been reached if it is satisfied. Since rather severe criteria are used, this remark is seldom in error. However, the procedure can terminate itself for other reasons, and it is these that must be remembered. If the maximum number of iterations is reached (normally 50, but easily altered by the user) or if the stress becomes satisfactorily small (normally 0.01, but easily altered by the user), or for several other reasons, the procedure will terminate with a suitable remark.

The reason for termination is always printed. You should always look at this comment. Even where the procedure does not state that a local minimum has been reached, it may often happen that the solution reached is practically indistinguishable from the local minimum that would be reached after a few more iterations. In particular, where the stress is very small, this is generally the case. The "history of the calculation" which may be printed out will often be helpful if convergence is in doubt.

With regard to local minima, there is no feasible way to be mathematically certain that you have found that local minimum which is in fact the true global minimum. (While use of the **TORSCA** initial configuration reduces the chance of reaching a local minimum in the sort of situations in which it has been tested, KYST offers so many possible modes of use that no universal reassurances can be given. In particular, unfolding is a relatively unexplored area.) However, by using a variety of different random starting configurations to repeatedly scale the same data, it is often possible to obtain practical certainty. For if a variety of different starting configurations yield essentially the same local minimum, with perhaps an occasional exception which yields a substantially worse local minimum, then there is little to worry about. One easy way to provide a variety of different starting configurations is to scale the same data repeatedly, using **SAVE=DATA** and **RANDOM=**integer, which will provide a new starting configuration on successive uses on the same computer run .

In this connection, it is well to remember that rotation and reflection leave a configuration essentially unchanged. Thus to decide whether the solutions obtained as suggested above are essentially the same, you cannot merely compare coordinates. One method you can use is to generate the matrix of distances, since these are unaffected by rotation and reflection. However, a still simpler method which is often adequate is to compare the stress values. If these values agree to three significant figures, the local minima having them are unlikely to be different. In the case of two-dimensional solutions, you can plot the solutions using the control phase **PLOT= CONFIGURATION=ALL**, trace them onto tracing paper, rotate or reflect the solutions by physical motions of the paper, and compare solutions by laying one sheet on top of another.

Note that the likelihood of reaching a local minimum is known by experience to be particularly severe for one-dimensional solutions, or if the Minkowski *r*-metric is used with *r* near 1 or ∞, though for ordinary scaling the **TORSCA** initial configuration greatly alleviates the situation. The one-dimensional case is bad because points cannot easily move past each other during the iterative computation if they lie on a line. The extreme values of *r* cause trouble because the gradient is a discontinuous function of the configuration if $r = 1$ or if $r = \infty$.

16. The Routines Which Make Up KYST

KYST is made up of a main program and 28 subroutines, all written in FORTRAN IV. The routines contain many comment cards which should prove helpful to the user who wishes to modify the program.

These are the routines:

KYST	(Main routine) Multidimensional scaling.
BLOCK DATA	Contains tables of control.
CCACT	Control cards action. Reads and interprets control cards.
DATAPR	Data print. Prints data and/or distances.
DTRAN	Data transformation. Used with **DFUNCTION** option.
WTRAN	Weights transformation. Used with **WFUNCTION** option.
FITM	Fit monotone. Performs least- squares monotone regression.
FITP	Fit polynomial. Performs least-squares polynomial regression.
REGR	Regression functions. Provides values of monomials (functions regressed over).
EQSOLV	Equation solver. Solves up to five simultaneous linear equations by simple equations.
INICON	Computes initial configuration using Young-Torgerson
CONFIG	Obtains a metric multidimensional scaling solution.
FITMS	Fit monotone. Performs least-squares monotone regression for initial configuration.
NEWSTP	New step. Calculates size of step to be taken along gradient.
PLOT	Generates a one-page printer plot of a matrix Y versus a vector X.
SERCH	Searches for next item to be processed
XITEM	Put X vector elements into the current print line.
RM1POW	Calculates the (R-1)th power.
RPOWER	Calculates the Rth power.
RROOT	Calculates the Rth root.

---

*We thank Dr. Peter Businger for providing this modern, efficient program which uses the deflation technique.

RUNIFV	Returns random uniform variable between 0 and 1. Inefficient* but portable.
SGEV	Eigenvector package. Returns
BSEC1	the k algebraically largest
DFLT	eigenvalues and eigenvectors
NRMLZ	of an NxN real symmetric
NVIT1	matrix given its upper trian-
SBK	gular part.
SORT	Sorting routine. Simple but efficient in-core sorting routine, using Shellsort technique.

17. Useful Information for Getting KYST Running at a Computer Center

All routines for KYST are written in FORTRAN IV. While considerable effort has been made to improve the portability of KYST, and while its predecessor routines M-D-SCAL and TORSCA are widely used, it is not possible to avoid all difficulties. We describe here the limitations on portability and the preparation which is needed to install KYST, to the best of our knowledge.

Some computer systems assign different amounts of space to real variables and integer variables. As KYST equivalencies certain real and integer arrays to each other, it will not work on such computer systems without appropriate action. Since KYST uses quite elaborate equivalencing of arrays and labelled common as shown in Fig. 3 at the end of this section , it is virtually a necessity to handle this .problem by some systematic method which maintains all the relative placements of arrays as shown in the figure. (These complex arrangements achieve substantial economy of space and efficient transmission of information among subroutines.)

It was not possible to avoid machine- and system-dependent constants in KYST. However, they are all grouped together in a data statement in the **BLOCK DATA** subroutine, and described here. It is vital that you set these seven values appropriately for your computing system! There are three machine-dependent constants (contained in a common block called **ACCUR**):

PRECSN = relative precision of floating-point numbers (present value is 1.5*10E-8);
XMAG = least possible strictly-positive number (present value is 1.0*10E-38);
XMAXN = greatest possible floating-point number, or the greatest possible absolute value of a negative floating-point number, whichever is smaller (present value is 1.0*10E38).

PRECSN should and **XMAG** may exceed the true value slightly, while **XMAXN** may be slightly less. The present values reflect a 36-bit machine word in which floating-point numbers use eight bits for exponent and 28 bits for mantissa and sign. There are four system-dependent constants (contained in a common block called **MDSCL1**):
LREAD = input file number (present value is five)
LPRINT = printer or output file number (present value is six);
LPUNCH = card punch output file number (present value is seven);
LSCRAT = file number for small amount of scratch space, used only by **CCACT** (present value is eight).

The word size requirements for KYST are not fully explored. We believe that positive integers need not exceed 216 - 1, and that negative integers are far more limited. Integers receive complex handling in three circumstances (where they serve as code numbers in **CCACT**, where three items per word are manipulated in the `IJ` array, and in the random number generator): in the first two circumstances integers do not exceed 215 - 1, in the last 215 - 1. As for the floating-point numbers, we believe that the accuracy required by KYST is quite low, but we have no concrete evidence.

The approximate storage space required on two different computer systems is shown below, where the Honeywell 6000 system is the one in use at the Murray Hill Computer Center of Bell Laboratories, and the IBM 370/165 at the Computer Center of the University of North Carolina.

Number of machine-words (Number of bytes) required for Honeywell 6000, IBM 370/165

All real and integer arrays	14,003
Program of KYST itself, and non- array data	11,689
Subtotal (Storage needed for the program)	25,692
Library subroutines, buffers, etc., loaded with KYST package by the monitor system	7,300
TOTAL	32,992

Common Storage Areas

All the labelled common blocks are listed below as they appear in each routine which uses them. Integer variables are marked with *, floating point variables with ., and equivalenced variables with =. The lengths of all arrays are indicated.

1. `ACCUR: PRECSN., XMAG., XMAXN.`
Used in: `MAIN, BLOCK DATA, INICON, PLOT, SERCH, SGEV.`

2. `MDSCL1: LREAD*, LPRINT*, LPUNCH*, LSCRAT*.`
Used in: `MAIN, BLOCK DATA, CCACT, DATAPR, FITM, FITMS, INICON, PLOT, SERCH.`

3. `METRIC: RTYPE*, RM1., RECR., RR.`
Used in: `MAIN, RPOWER, RROOT, RM1POW.`

4. `PLTCHR: PTID 100., ITEM 101.`
Used in: `MAIN, BLOCK DATA, PLOT, XITEM.`

5. `KYST1: DATA 1800.,WW 1800.,IJ 1800*,X 600.`
Used in: `MAIN, CONFIG, DATAPR, FITM, FITP, INICON.`

6. `MDSCL3: As used in BLOCK DATA: MTAB 13*, TAB1 64*, TAB2 68*, TAB3 64*, TAB4 76*, TAB5 72*.`
As used in `CCACT: MTAB 13*, TAB 344* = ATAB 344.`

7. `MDSCL2:` As used in `MAIN: LDIMX*, LDIMN*, LDIMD*, CUTOFF., STRMIN., SFGRMN., COSAVW., ACSAVW., IFIRST*, MATSWP*, SRATST., LSCH*, LPUNSW*, LSPL*, LRANDC*, LDAPRT*, LDIPRT*, LREG*, LHIPRT*, NUDASW*, LPOLSP*, LCONSW*, LNFIXZ*, DCON1., DCON2., DCON3., DCON4., DCON5., WCON1., WCON2., WCON3., WCON4., WCON5., NOITIN*, LPLCON*, LPLSCT*, LCOORS*.`
As used in `BLOCK DATA: LPAR 42* = PAR 42.`
As used in `DTRAN: DUMMY 28., A., B., P., S., T., WCON1., WCON2., WCON3., WCON4., WCON5., DUM 4.`
As used in `WTRAN: DUMMY 28., DCON1., DCON2., DCON3., DCON4., DCON5., A., B., P., S.,T., DUM 4.`

8. `KYST2:` diagram on next page

Figure 3

References

Coombs, C. (1964) *A Theory of Data*. New York: John Wiley & Sons, 1964. Chapters 5, 6, and 7, pages 80-180.

Klahr, D. (1969) A Monte Carlo investigation of the statistical significance of Kruskal's nonmetric scaling procedure. *Psychometrika*, 1969, **34**, 319-330.

Kruskal, J. B. (1964a) Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 1964, **29**, 1-27.

Kruskal, J. B. (1964b) Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 1964, **29**, 115-129.

McGee, V. E. (1968) Multidimensional scaling of N-sets of similarity measures. A nonmetric individual difference (?) *Multivariate Behavioral Research*,

Shepard, R. N. (1962) The analysis of proximities: Multidimensional scaling with an unknown distance function. *Psychometrika*, 1962, **21**, 125-139, 219-246.

Sherman, C. R. (1972) Nonmetric multidimensional scaling: A Monte Carlo study of the basic parameters. *Psychometrika*, 1972, **51**, in press.

Spence, I. A., & Ogilvie, J. C. A table of expected stress values for random rankings in nonmetric multi-dimensional scaling. *Multivariate Behavioral Research*, in press.

Spence, I. A. (1972) A Monte Cario evaluation of three nonmetric multidimensional scaling algorithms. *Psychometrika*, 1972, **31**, in press.

Stenson, H. H., & Knoll, R. L. (1969) Goodness of fit for random rankings in Kruskal's nonmetric scaling procedure. *Psychological Bulletin*, 1969, **11**, 122-126.

Wagenaar, W. A., & Padmos, P. (1971) Quantitative interpretation of stress in Kruskal's multidimensional scaling technique, *British Journal of Mathematical and Statistical Psychology*, 1971, **24**, 10l-110.

Young, F. W. (1970) Nonmetric multidimensional scaling: recovery of metric information. *Psychometrika*, 1970, **35**, 455-473.

Young, F. W. (1972) A model for polynomial conjoint analysis algorithms. In R. N. Shepard, A. K. Romney, & S. B. Nerlove (Eds.), *Multidimensional scaling: Theory and applications in the behavioral sciences*. New York Seminar Press, 1972 Vol 1, pages 9-102

Index of Control Phrases

(Underlined phrases are default options and numbers in parentheses are default values.)

Word or Phrase & Page

ABSENT 12
ACSAVW=decimal number (.66) 25
ARBITRARY 19,20
ASCENDING 21
AS-IS 26
BLOCKDIAGONAL=NO 12,13
=YES 12,13
BYDECKS 22
BYGROUPS 22
BYROWS 22
CARDS 25
COMPUTE 6,11
CONFIGURATION 19,20,26
CONSTANT 22
COORDINATES=AS-IS 26
=STANDARDIZE 26
=ROTATE 26
COSAVW=decimal number (.66) 24
CUTOFF=decimal number -1.23*10E20 13
DCONSTANTS=five decimal numbers (0.0, 1.0, 1.0, 0.0, 1.0) 14
DESCENDING 22
DFUNCTION 14
DIAGONAL=ABSENT 12
=PRESENT 12
DIMMAX=integer (2) 21
DIMDIF=integer (1) 21
DIMMIN=integer (2) 21
DISTANCES 26
FIX=integer (0) 23
HISTORY 25
ITERATIONS=integer (50) 23,24
LOWERCORNERMATRIX 12,13
LOWERHALFMATRIX 12,13
MATRIX 12
MULTIVARIATE 22
NO 12,13
NOCARDS 25
NOCONSTANT 22
NODATA 25
NODISTANCES 26
NOHISTORY 25
NOMORE 22,23
NONE 30
PLOT=CONFIGURATION=ALL 26
=SOME 26
=NONE 26
PLOT=SCATTER=ALL 26
=SOME 26
=NONE 26
POLYNOMIAL 22
PRESENT 12

PRE-ITERATIONS=integer (1) 19
PRIMARY 21
PRINT=DATA 25
    =NODATA 25
    =DISTANCES 26
    =NODISTANCES 25
    =HISTORY 25
    =NOHISTORY 25
R=decimal number (2.0) 21
RANDOM=integer (0) 19,20
REGRESSION=ASCENDING 22
    =DESCENDING 22
    =POLYNOMIAL=integer 22
    =CONSTANT 22
    =NOCONSTANT 22
    =MULTIVARIATE=integer 22
ROTATE 26
SAVE=CONFIGURATION 19,20
    =DATA 13
SCATTER 26
SECONDARY 21
SFGRMN=decimal number (0.0) 24
SFORM1 22
SFORM2 22
SOME 26
SPLIT=BYDECKS 22,23
    =BYGROUPS 22,23
    =BYROWS 22,23
    =NOMORE 22,23
SRATST=decimal number (.999) 24
STANDARDIZE 26
STOP 11
STRMIN=decimal number (.01) 24,25
TORSCA 8,19
UPPERHALFMATRIX 12,13
UPPERCORNERMATRIX 12,13
WCONSTANTS=five decimal numbers (0.0, 1.0, 1.0, 0.0, 1.0) 15
WFUNCTION 15
YES 12,13