A computer program that tests for the compatibility of two multidimensional point sets is described. The program provides a measure of the discrepancy between the two point distributions, as well as giving information concerning the regions of the multidimensional space where the discrepancies occur. A procedure for calculating the significance level of the test in any given situation is also provided.
INTRODUCTION

COMPAR is a collection of subroutines that implement the multidimensional point set comparison algorithm of Friedman, Steppel, and Tukey. The algorithm and its properties are detailed in references 1 and 2 and are only discussed briefly here as they relate to its implementation.

Consider two samples of $N_1$ and $N_2$ observations, taken on vector random variables $\vec{x}$ and $\vec{y}$ with unknown probability density functions $f(\vec{x})$ and $g(\vec{y})$, defined over a multidimensional space of dimensionality, $d$. We wish to test the hypothesis (called the null hypothesis, $H_0$) that $f(\vec{x}) = g(\vec{y})$ for all $\vec{x}$ and $\vec{y}$.

PROCEDURE

The algorithm for testing this hypothesis proceeds as follows: the two samples (classes) of size $N_1$ and $N_2$, respectively, are combined into a single sample of size $N = N_1 + N_2$. Each point is tagged as to the class from which it came. The closest $k$-neighbors to each point are examined and the number, $k_1$, that originate from class one is determined. Thus, associated with each point in this combined sample is this measure, $k_1$, of the composition of the points closest to it. The quantity $k_1$ is then histogrammed separately for the class one points and the class two points. These histograms are then compared to the distributions expected when the two multidimensional point sets are compatible.

It is shown in Reference 1 that this expected distribution for the histogram of $k_1$ values, for both the class one and class two points, is a binomial distribution over $k_1 = 0, 1, 2, 3, \ldots, k$ with probability $N_1/N$. That is,

$$f(k_1) = \binom{k}{k_1} p^{k_1} (1-p)^{k-k_1}$$

with

$$p = \frac{N_1-1}{N-1}$$

for the class 1 points,
and \( p = \frac{N_1}{N-1} \) for the class 2 points.

The two histograms of \( k_1 \) values for the class one and class two points are each separately compared to \( f(k_1) \) (eqn 1) using a univariate (one-dimensional) goodness-of-fit test analogous to a Pearson's \( \chi^2 \) test statistic. That is,

\[
T_1 = \sum_{k_1=0}^{k} \frac{[n_1(k_1) - f(k_1)M_1]^2}{f(k_1)M_1}
\tag{2a}
\]

for the class one histogram and

\[
T_2 = \sum_{k_1=0}^{k} \frac{[n_2(k_1) - f(k_1)M_2]^2}{f(k_1)M_2}
\tag{2b}
\]

for the class two histogram, where \( n_1(k_1) \) represents the histogram values for the class one points and \( n_2(k_1) \) represents the histogram values for the class two points. \( M_1 \) and \( M_2 \) are the numbers of class one and two points, respectively, for which \( k_1 \) is evaluated and histogrammed. These numbers may be less than the full data sample sizes \( N_1 \) and \( N_2 \). A combined \( \chi^2 \) is then formed for the complete test by adding those for the two histograms,

\[
T = T_1 + T_2 \quad .
\tag{3}
\]

A log binned likelihood goodness-of-fit test is also used:

\[
w_1 = \log(N_1!) + \sum_{k_1=0}^{k} n_1(k_1) \log f(k_1) - \log[n_1(k_1)!] \tag{4a}
\]

\[
w_2 = \log(N_2!) + \sum_{k_1=0}^{k} n_2(k_1) \log f(k_1) - \log[n_2(k_1)!] \tag{4b}
\]
and

\[ w = w_1 + w_2 \]  \hspace{1cm} (5)

The quantity \( \lambda = -2w \) is usually used as the test statistic.

In order for this test to be useful, the significance level, \( \alpha(T) \), (confidence level, P-value) for an obtained experimental value of the test statistic (\( T \) or \( \lambda \)) must be determined. This significance level is the probability that one would observe a larger value of \( T \) or \( \lambda \) than that experimentally observed if the null hypothesis were true, (i.e., the two multidimensional point sets were compatible).

Unfortunately, this significance level cannot be simply evaluated from the usual assumption that \( T \) and \( \lambda \) follow a chi-squared distribution with \( 2k \) degrees of freedom. This is due to the lack of independence in the measurements of the \( k_1 \) values appearing in each histogram. Because the nearest neighborhoods of each point in the multidimensional space are not mutually exclusive but overlap considerably, the values of \( k_1 \) for neighboring events are highly correlated. These correlations cause the distribution of values of the test statistic to deviate considerably from that when the values of the histogrammed quantity are all independent. This deviation usually takes the form of an increased spread of the \( T \)-distribution. That is, the mean value stays the same as for the independent case, \( \bar{T} \approx 2k \), but its spread about that mean value is considerably wider than a \( \chi^2 \)-distribution with \( 2k \)-degrees of freedom.

Another problem is that the \( T \)-distribution can also depend upon the specific underlying distributions of the multidimensional point sets. This is a usual property of multidimensional statistical procedures. Thus, for each different problem the \( T \) or \( \lambda \) null-distribution could, in principle, be different. However, for this test, these distributions are remarkably independent of the underlying multidimensional point distributions.
It is possible to overcome these problems by employing a permutation procedure to estimate the significance level of the test directly from the data for each application. This permutation test proceeds as follows: the two samples (classes) are combined into a single sample of size $N = N_1 + N_2$, as described above. But instead of assigning each sample point to the class from which it originated, it is randomly assigned to one of the two classes. These random assignments are made subject to the constraint that the assignments preserve the original proportion $N_1/N_2$. The comparison procedure is then applied to these two newly defined samples and a test statistic value (eqns 3 or 5) is obtained. The points are then given another such random assignment and the test re-applied, obtaining another permuted test statistic value. Repeated application of this random permutation procedure yields a series of test statistic values that closely approximate the null test statistic distribution for the given problem. In particular, the fraction of these permuted test statistic values that are larger than the value, $T$ (or $\lambda$), obtained for the unpermuted case, is an estimate of the significance level, $\alpha(T)$, for the test.

This comparison procedure can also be used to identify those regions of the multidimensional space where the two point samples most disagree (or agree) in their relative densities. This is because the algorithm assigns such an estimate to each point in the combined sample. Those points for which $k_1$ is near its expected mean value, $k_1 = pk$ are located in regions where the agreement is good, while those points for which $k_1$ is far from this expected mean value are located in regions of the multidimensional space where the agreement is bad. One can use the experimentally determined values of $k_1$ for each point to select and isolate those points that give rise to the most disagreement (or agreement) between the two point sets. The coordinates of these selected points can then be examined to determine where these points lie in the multidimensional space.
This procedure leaves to the researcher's discretion the choice of the coordinate variables and metric, and the number of nearest neighbors, k. This procedure is reasonably insensitive to the choice of k, provided that it is not too small. In order for the test to be consistent, k should be a function of the total sample size such that

$$\lim_{N \to \infty} k(N) = \infty$$, and $$\lim_{N \to \infty} \frac{k(N)}{N} = 0.$$ 

Experimentation has shown that the choice of k is not important so long as k > 10-20. Clearly, k should be small compared to the total sample size, N.

This algorithm is somewhat more sensitive to choice of measurement variables and metric. Unfortunately, there are no good guidelines for their choice. For very large sample size, the algorithm is clearly invariant to changes in coordinate variables and metric since these changes simply alter the shape of the volume element containing the evaluation point. Since these volumes are infinitesimally small, their shape doesn't matter.

For finite sample sizes, however, the shape does matter. Changes in the volume shape that result in changes of the identities of the nearest neighbors will have an effect on the performance of the algorithm. Fukunaga and Hostetler\(^3\) show that for those data distributions that can be made spherically symmetric by a linear transformation, the optimum metric is the inverse covariance matrix of the underlying distribution, \(p(x)\). If this covariance matrix is estimated by the data sample covariance matrix, then this is equivalent to scaling each of the coordinates so that they have equal variance along the principal axes of the data (sphericalizing the data).

If there is no apriori information concerning the data, then this is probably the best procedure. Another reasonable procedure is to simply scale the
data to have equal variance along the original measurement coordinates. On the other hand, different experimental measurement accuracy or different characteristic length of density variation can dictate unequal scales among the various coordinates. Changing the scale of a coordinate changes its relative importance in determining the goodness-of-fit. Thus, if the researcher has information as to which coordinates are most important, they should be given larger scales.

The number and specific choice of coordinate variables also affect the performance of this procedure. Increasing the number of coordinates only improves the performance when those variables contain information concerning the hypothesis under test. In fact, coordinates that do not contain such information (noise coordinates) dilute the power of the test. This is because these dimensions add statistical uncertainty to the estimates without providing information helpful to the comparison. Even a coordinate that does contain some additional information may not help because the increase in statistical variance that it introduces hurts more than the information increase helps. The precision of these tests can be increased greatly if the researcher's knowledge and intuition lead him to a judicious choice of coordinate variables.

After the specific measurement variables and their scales are chosen, there is still the choice of the metric or multidimensional dissimilarity measure. That is, given a one-dimensional dissimilarity measure for each coordinate, how are these combined to define a distance in the multidimensional space. One-dimensional distance is nearly always defined as the absolute value of the difference in coordinate values

\[ d_{mn}^{(i)} = | x_m^{(i)} - x_n^{(i)} | , \]  

where the subscripts label the points and the superscripts label the coordinate. The most commonly used multidimensional distance measures are the Minkowski
p-metrics,

\[ d_p(\mathbf{x}_m, \mathbf{x}_n) = \left[ \sum_{i=1}^{d} |x_m^{(i)} - x_n^{(i)}|^p \right]^{\frac{1}{p}} \]  

Of these, the three most popular are the

- \( p=1 \) (city block or taxi cab distance)
- \( p=2 \) (Euclidean distance measure)
- \( p=\infty \) (maximum axis distance measure)

i.e.,

\[ d_{\infty}(\mathbf{x}_m, \mathbf{x}_n) = \max_{1 \leq i \leq d} \{|x_m^{(i)} - x_n^{(i)}|\} \]  

The advantage of the \( p=1 \) measure is that it can be calculated rapidly with no multiplications. It is also relatively robust against changes in relative axis scales. The advantage of the \( p=2 \) distance measure is that it is the natural extension of the familiar Euclidean distance measure used in plane and solid geometry. It is somewhat less robust to changes in axis scales than the \( p=1 \) metric. The \( p=\infty \) metric can be calculated rapidly with no multiplication, like the \( p=1 \) metric, and has the additional advantage that the nearest neighbors can be found much more rapidly for this distance measure (especially for high dimensionality) than with the other two metrics. This metric has the greatest sensitivity to the relative coordinate scales.

The best choice for a distance measure depends upon the problem at hand, and is related to the underlying density distribution of the multidimensional data. As for the cases of choice of variables and scales, there are no good guidelines. It can be shown that the \( p=2 \) metric is optimal when the data points are distributed according to a multidimensional Gaussian distribution. However, there are no general results for other types of distributions. COMPAR offers the choice of \( p=2 \), since it is the one most commonly used, and \( p=\infty \) because of the high efficiency it affords in the nearest neighbor calculation.
THE K-NEAREST NEIGHBOR CALCULATION

Almost the entire computational cost of employing this procedure is in the calculations required for finding the k-nearest neighbors to each sample point. The most straightforward procedure is to simply calculate the distance from each point to all of the other points and record the k-smallest. This is known as the brute force (BF) method and requires computation proportional to the dimensionality, $d$, and proportional to the square of the sample size,

$$C_{BF}(d,k,N) = \alpha_{BF}(k) d N^2. \quad (8a)$$

Recently, several new algorithms for finding nearest neighbors have been reported$^{4,5}$ that are faster than the brute force method for sufficiently large $N$ and small $d$. The method of Friedman, Baskett, and Shustek (FBS)$^4$ finds the k-nearest neighbors with computation

$$C_{FBS}(d,k,N) = \alpha_{FBS} \left[ \frac{kd \Gamma \left( \frac{d}{2} \right)}{2} \right]^{1/d} N^2 - \frac{1}{d} \quad (8b)$$

while the method of Bentley (B)$^5$ finds them with computation

$$C_B(d,k,N) = \alpha_B(k) 2^d N \log_2 N \quad (8c)$$

The advantage of the BF-method is its low overhead requirements in both computation per distance calculation and additional memory. The FBS method introduces a small amount of computational overhead per distance calculation and a substantial increase in memory requirements (depending upon the optimization level used in the algorithm). The B-algorithm introduces a large computational overhead per distance calculation and requires an amount of additional memory comparable to the FBS method.

The best algorithm to use depends upon the combined sample size, $N = N_1 + N_2$, the dimensionality, $d$, to a smaller extent the number of near neighbors, $k$, and on the amount of memory available. The FBS method is seldom
slower than the BF-method and, in addition, it can calculate when it will be slower and automatically transfers to the BF-method for those cases. Therefore, if memory size is not a restriction, the FBS method should always be favored over the BF method.

For small to moderate sample size ($N \leq 2500$), the FBS method is also considerably faster than the B-method. However, for very large sample sizes the B-method becomes much faster. The actual sample size at which the B-algorithm surpasses the FBS-algorithm in performance depends strongly on the dimensionality, $d$, and is tabulated in Table 1 for various values of $d$.

The additional storage required by the FBS method is $s(\text{FBS}) = l (N+2)+d$ where $l (1 \leq l \leq d)$ is the optimization level used with the algorithm. Higher optimization levels give better results but require more storage. The additional storage required by the B-algorithm is $s(\text{B}) = 3N$.

**IMPLEMENTATION**

The multidimensional point set comparison algorithm is invoked from a FORTRAN program by a subroutine call

```
CALL COMPAR(D,N1,N2,M,X,K,METRIC,FLAG,L,STORE,C).
```

All quantities are input to COMPAR except C.

- **D** = number of attributes per data point (dimensionality of data space).
- **N1** = number of class one sample points.
- **N2** = number of class two sample points.
- **M** = number of sample points for which the $k$-nearest neighbors are found, $k$-evaluated and histogrammed. The quantity $M$ must be less than or equal to $N=N1+N2$. The M subsample is drawn randomly from the total input data sample.
\[ X = \text{input data array dimensioned } X(D,N1+N2+1) \]
\[ X(1,1) - X(D,N1) \text{ contain the class one data points} \]
\[ X(1,N1+1) - X(D,N1+N2) \text{ contain the class two data points} \]
\[ X(1,N2+N2+1) - X(D,N2+N2+1) \text{ are scratch locations used by COMPAR.} \]

\[ K = \text{number of nearest neighbors used.} \]

\[ \text{METRIC} = \begin{cases} 'P1WO' \text{ or 'ECLD' use } p=2 \text{ (Euclidean) metric.} \\ 'PINF' \text{ or 'MAXA' use } p=\infty \text{ (maximum axis) metric.} \end{cases} \]

\[ \text{FLAG} = \begin{cases} 'USER' \text{ compare original two data samples} \\ 'SHFL' \text{ randomly reassign (shuffle) identities then compare} \\ 'BOTH' \text{ do both of the above} \end{cases} \]

COMPAR can be called repeatedly with \[ \text{FLAG} = 'SHFL'. \] Each such call will produce a new random reassignment of the point identities, and then a comparison. Thus, repeated calls to COMPAR with \[ \text{FLAG} = 'SHFL' \] will produce a series of test statistic values that can be compared with the \[ \text{FLAG} = 'USER' \] value. The fraction of the \[ \text{FLAG} = 'SHFL' \] test statistic values larger than the \[ \text{FLAG} = 'USER' \] value is an estimate of the significance level of the test.

\[ L = \text{optimization level for k-nearest neighbor calculation.} \]

\[ L = \begin{cases} 0 \text{ use BF method} \\ 1-D \text{ use FBS method at optimization level } L \\ D \text{ use B-method} \end{cases} \]

\[ \text{STORE} = \text{array dimensioned } \text{STORE}(L*(N1+N2+2)) \]

\[ \text{for } 1 \leq L \leq D \text{ (FBS algorithm),} \]
\[ \text{ignored for } L=0 \text{ (BF method).} \]

\[ \text{If } L \geq D \text{ (B-algorithm) then} \]

\[ \text{dimensioned } \text{STORE}(3*(N1+N2)) \]

\text{NOTE: The memory requirements for the STORE array listed here are those required to execute the program on an IBM 360/370 computer. Internally to COMPAR, the STORE array is used for pointing to the data elements in the input array and is treated as an INTEGER*2 array (16 bits). This imposes an upper limit on the total sample size of } N = N_1 + N_2 \leq 32767. \] If a larger total sample size is required, or if COMPAR is executed on a computer that does not allow a facility for half-word integers, then the memory required for STORE is twice that listed here, and all INTEGER*2 declarations internal to COMPAR must be changed to INTEGER.
\[ C \quad = \quad \text{real array dimensioned } C(N_1+N_2,2) \]

\[ C(i,l) = \text{experimentally determined value of } k_1 \]

for \( C(i,2) \)-th data point for \( l \leq i \leq M \).

That is, \( C(i,2) \) refers to the position of the data point in the original input list and \( C(i,l) \) contains the number of class one data points in its \( k \)-nearest neighborhood.

**COMMON /COMPO/CHI,ALAMB,TSQ**

Upon return from COMPAR, CHI contains the value of the combined chi-squared, \( T \), (eqns 2-3), ALAMB contains combined \( \lambda = -2w \) (eqns 4-5), and TSQ contains the value of an additional test statistic

\[ t^2 = \sum_{h=1}^{2} \frac{[k_1(h) - p_k]^2}{(k_1^2 - k_1^2(h))}. \quad (9) \]

Here the sum is over the two histograms of \( k_1 \) values. \( k_1 \) is the experimental mean value of \( k_1 \) for the histogram, \( p_k \) is the expected mean value under the null hypothesis (see eqn 1) and the denominator is the variance of the \( k_1 \) distribution for the histogram.

**DATA SPHERICALIZATION**

As discussed above, if there is no information to the contrary, all the coordinates (and all linear combinations of the coordinates) should be given equal importance. This is accomplished by transforming the entire data set so that it has equal variance about all of its principal axes. The covariance matrix for the transformed data is then a constant and cannot be changed by rotating the coordinate system.

If the input data set contains one (or several) linear constraints, then the data will have zero variance about one (or several) of its principal axes.
For this case, the data lie on a lower dimensional linear manifold embedded in the full \( d \)-dimensional space. Since the spheroidalization procedure identifies these hyperplanes, it can reduce the dimensionality of the data set without any loss of information. As pointed out above, this will always increase the sensitivity of the algorithm.

Data spheroidalization is accomplished with a subroutine call

\[
\text{CALL SPHERE}(D,N,X,Y,E).
\]

All quantities are input except \( Y \). \( D,N \), and \( X \) have the same meanings as above.

\[
Y = \text{array dimensioned the same as } X. \text{ Upon return, it contains the data points transformed to the spheroidal representation. The } Y\text{-array may be identical to the } X\text{-array, in which case the original data will be overwritten by the transformed data.}
\]

\[
E = \text{Real*8 array dimensioned Real*8 } E(D,D,2). \text{ It is a scratch array for } \text{SPHERE.}
\]

The transformed data, \( Y \), can then be used as input to \text{COMPAR}.

\text{Labeled Common}

\text{COMMON/EITMIN/EFACT}

The quantity \( \text{EFACT} \) controls the reduction of the linear dimensionality due to linear constraints. Specifically, a principal axis is considered a constraint direction if its eigenvalue \( \lambda \) satisfies the relation

\[
\frac{\lambda}{\lambda_{\max}} < \text{EFACT}
\]

where \( \lambda_{\max} \) is the largest principal eigenvalue. The value of \( \text{EFACT} \) is set to 0.01 in a block data subprogram internal to \text{SPHERE}. This value may be changed at the user's discretion.
The nature and amount of printed output from the program depends upon the options chosen by the user. Figure 2 displays the printed output arising from the program shown in Figure 1.

The first two lines are output from SPHERE, and appear only if this subroutine is called. The first line (BASIC LINEAR DIMENSIONALITY = ) lists the dimensionality, d, minus the number of linear constraints found in the sphericalization procedure. The next line lists the scale factors (divisors) used to scale the data along each of its principal axes.

Following this output from SPHERE is the output from COMPAR. The first line lists the number of near neighbors, k, and the metric chosen by the user. The next line lists the sample sizes (N_1 and N_2) of the input data. The next line indicates whether the comparison is made on the user input data or whether the data has been randomly permuted (shuffled) before the comparison was made.

The following line lists the number of data points, M, for which the nearest neighbors are found, and k_1 evaluated and histogrammed. This number is divided into the number of Class 1 points, M_1, and Class 2 points, M_2, with M = M_1 + M_2. Since the M evaluation points are randomly drawn from the N = N_1 + N_2 total sample points, the ratio M_1/M_2 will not generally be exactly equal to the ratio N_1/N_2 in the total sample.

The following lines of printed output present the results of the comparison. The first line gives various statistical averages for the histogram of the M_1-Class 1 evaluation points and the next line for the M_2-Class 2 evaluation points. These quantities are the experimental value of \( \bar{k}_1 \) (AVERAGE = ), expected value of \( \bar{k}_1 \) for null hypothesis (EXPECT = ).
\[ E_1[k_1] = \frac{N_1-1}{N-1} k \] for class 1 histogram,

\[ E_2[k_1] = \frac{N_1}{N-1} k \] for class 2 histogram

and the experimental value for \( \bar{k}_1^2 - \bar{k}_1^2 \) (VARIANCE = ), and its expected value for the null hypothesis:

\[ E[(k_1 - \bar{k}_1)^2] = E_1[k_1] \left( 1 - \frac{N_1 - 1}{N-1} \right) \] for class 1 points

\[ = E_2[k_2] \left( 1 - \frac{N_1}{N-1} \right) \] for class 2 points.

The estimates of the uncertainties in these experimental quantities, listed after them (+/-), are calculated assuming that all the \( k_1 \) measurements appearing in the histograms are independent. As discussed above, these quantities have a positive correlation so that the actual statistical uncertainty can be somewhat larger than those listed. The uncertainties listed represent a lower limit on the true statistical uncertainties. For the case where \( M < < N \), then the true uncertainties are very close to those listed, but when \( M \sim N \), the true uncertainties can be up to twice as large as those listed.

The next line prints the values of the test statistics, \( T \), Eqn 3, (CHISQUARED = ), \( \lambda \), Eqn 5, (LAMBDAA = -2*W = ), and \( t^2 \), Eqn 9, (T-SQUARED = ) resulting from the comparison.

Following this is a pictorial representation of the histograms of \( k_1 \) values for the class 1 and class 2 evaluation points separately. The total content of these histograms is \( M_1 \) and \( M_2 \), respectively. The expected number of counts in each bin (for the null hypothesis) are represented by *'s superimposed over the histogram. The pictorial representation plots the square roots (rootogram) of both the experimental and expected numbers of counts. In
this representation, the expected statistical deviations of the experimental
counts from the null hypothesis (when the null hypothesis is true), are con-
stant.

Following the pictorial rootogram representation, the expected number of
counts in each bin for the null hypothesis (represented by *'s in the rooto-
gram) are explicitly listed.

For subsequent calls to COMPAR, this complete output sequence is printed
for each such call. If FLAG is set to 'BOTH' (as in the example of Figure 1),
then the input information is not repeated for the second (data permuted) com-
parison.
REFERENCES


6. See Reference 2, pp 54.
Sample size, $N$, for various dimensionalities, $D$, at which the Bentley (B) algorithm for finding nearest neighbors equals the performance of the Friedman, Baskett, Shustek (FBS) algorithm. For smaller sample sizes, the FBS method is faster while for larger sample sizes, the B algorithm is faster. All trials were done with 20 nearest neighbors. For a larger number of nearest neighbors, the table entry is somewhat lower while for a smaller number, the entry is larger. In all cases, optimization level equal to the dimensionality was used with the FBS method. All trials were run on an IBM 370/168 with FORTRAN H optimization level two compiler. For other computers and other compilers, these numbers should serve as reasonable approximations. The running times are for the corresponding dimensionality and total sample sizes. For other total sample sizes, the formulae in the text (eqn 8b and 8c) can be used to predict running times. If the number of evaluation points, $M$, is less than the total sample size, $N$, then all time estimates should be multiplied by the ratio $M/N$. For dimensionalities greater than ten, the FBS method should always be used.

<table>
<thead>
<tr>
<th>Dimensionality $D$</th>
<th>Cross Over Sample Size $N$</th>
<th>Running Time $T$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3900</td>
<td>25.6</td>
</tr>
<tr>
<td>3</td>
<td>2500</td>
<td>22.3</td>
</tr>
<tr>
<td>4</td>
<td>2700</td>
<td>46.0</td>
</tr>
<tr>
<td>5</td>
<td>2800</td>
<td>95.2</td>
</tr>
<tr>
<td>6</td>
<td>3900</td>
<td>409.6</td>
</tr>
<tr>
<td>7</td>
<td>5500</td>
<td>607.4</td>
</tr>
<tr>
<td>8</td>
<td>7750</td>
<td>1402.6</td>
</tr>
<tr>
<td>9</td>
<td>12000</td>
<td>3743.2</td>
</tr>
</tbody>
</table>
TWO SAMPLE POINT SET COMPARISON EXAMPLE.
FIVE DIMENSIONS.
1000 CLASS 1 POINTS AND 1000 CLASS 2 POINTS, 700 EVALUATION POINTS,
20 NEAR NEIGHBORS. EUCLIDEAN DISTANCE MEASURE.

REAL DATA(5,2001),C(4000),STORE(5,2002)
REAL*8 E(5,5,2)

CALL DATAIN(DATA,5,1000) READ IN CLASS 1 DATA.
CALL DATAIN(DATA(1,1001),5,1000)
CALL SPHERE(5,2000,DATA,DATA,E) SPHERICALIZE TOTAL DATA SAMPLE.
CALL COMPAR(5,1000,1000,700,DATA,20,'ECLD','BOTH',5,STORE,C) COMPARE POINT SETS.

STOP
END

FIGURE 1