NAG Toolbox for MATLAB

g03fc

1 Purpose

g03fc performs non-metric (ordinal) multidimensional scaling.

2 Syntax

```
[x, stress, dfit, ifail] = g03fc(typ, n, d, x, iter, iopt, 'ndim', ndim)
```

3 Description

For a set of n objects, a distance or dissimilarity matrix D can be calculated such that d_{ij} is a measure of how 'far apart' the objects i and j are. If p variables x_k have been recorded for each observation this measure may be based on Euclidean distance, $d_{ij} = \sum_{k=1}^{p} \left(x_{ki} - x_{kj}\right)^2$, or some other calculation such as the number of variables for which $x_{kj} \neq x_{ki}$. Alternatively, the distances may be the result of a subjective assessment. For a given distance matrix, multidimensional scaling produces a configuration of n points in a chosen number of dimensions, m, such that the distance between the points in some way best matches the distance matrix. For some distance measures, such as Euclidean distance, the size of distance is meaningful, for other measures of distance all that can be said is that one distance is greater or smaller than another. For the former metric scaling can be used, see g03fa, for the latter, a non-metric scaling is more appropriate.

For non-metric multidimensional scaling, the criterion used to measure the closeness of the fitted distance matrix to the observed distance matrix is known as **stress**. **stress** is given by,

$$\sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\left(\hat{d}_{ij} - \tilde{d}_{ij}\right)^2}{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\hat{d}_{ij}^2}}$$

where $\hat{d}_{ij}^2$ is the Euclidean squared distance between points i and j and $\tilde{d}_{ij}$ is the fitted distance obtained when $\hat{d}_{ij}$ is monotonically regressed on d_{ij}, that is $\tilde{d}_{ij}$ is monotonic relative to d_{ij} and is obtained from $\hat{d}_{ij}$ with the smallest number of changes. So **stress** is a measure of by how much the set of points preserve the order of the distances in the original distance matrix. Non-metric multidimensional scaling seeks to find the set of points that minimize the **stress**.

An alternate measure is squared **stress**, $sstress$,

$$\sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\left(\hat{d}_{ij}^2 - \tilde{d}_{ij}^2\right)^2}{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\hat{d}_{ij}^4}}$$

in which the distances in **stress** are replaced by squared distances.

In order to perform a non-metric scaling, an initial configuration of points is required. This can be obtained from principal co-ordinate analysis, see g03fa. Given an initial configuration, g03fc uses the optimization function e04dg to find the configuration of points that minimizes **stress** or $sstress$. The function e04dg uses a conjugate gradient algorithm. g03fc will find an optimum that may only be a local optimum, to be more sure of finding a global optimum several different initial configurations should be used; these can be obtained by randomly perturbing the original initial configuration using functions from Chapter G05.

4 References

Chatfield C and Collins A J 1980 *Introduction to Multivariate Analysis* Chapman and Hall

Krzanowski W J 1990 *Principles of Multivariate Analysis* Oxford University Press

5 Parameters

5.1 Compulsory Input Parameters

1: **typ – string**

Indicates whether **stress** or $sstress$ is to be used as the criterion.

typ = 'T'

stress is used.

typ = 'S'

$sstress$ is used.

Constraint: **typ** = 'S' or 'T'.

2: **n – int32 scalar**

n, the number of objects in the distance matrix.

Constraint: **n** > **ndim**.

3: **d(n × (n − 1)/2) – double array**

The lower triangle of the distance matrix D stored packed by rows. That is $\mathbf{d}((i-1) \times (i-2)/2 + j)$ must contain d_{ij}, for $i = 2, 3, \ldots, n$ and $j = 1, 2, \ldots, i-1$. If d_{ij} is missing then set $d_{ij} < 0$; for further comments on missing values see Section 8.

4: **x(ldx,ndim) – double array**

ldx, the first dimension of the array, must be at least **n**.

The ith row must contain an initial estimate of the co-ordinates for the ith point, for $i = 1, 2, \ldots, n$. One method of computing these is to use g03fa.

5: **iter – int32 scalar**

The maximum number of iterations in the optimization process.

iter = 0

A default value of 50 is used.

iter < 0

A default value of $\max(50, 5nm)$ (the default for e04dg) is used.

6: **iopt – int32 scalar**

Selects the options, other than the number of iterations, that control the optimization.

iopt $= 0$

Default values are selected as described in Section 8. In particular if an accuracy requirement of $\epsilon = 0.00001$ is selected, see Section 7.

iopt > 0

The default values are used except that the accuracy is given by 10^{-i} where $i =$ **iopt**.

iopt < 0

The option setting mechanism of e04dg can be used to set all options except **Iteration Limit**; this option is only recommended if you are an experienced user of NAG optimization functions. For further details see e04dg.

5.2 Optional Input Parameters

1: **ndim – int32 scalar**

Default: The dimension of the array **x**.

m, the number of dimensions used to represent the data.

Constraint: **ndim** ≥ 1.

5.3 Input Parameters Omitted from the MATLAB Interface

ldx, wk, iwk

5.4 Output Parameters

1: **x(ldx,ndim) – double array**

The ith row contains m co-ordinates for the ith point, for $i = 1, 2, \ldots, n$.

2: **stress – double scalar**

The value of **stress** or $sstress$ at the final iteration.

3: **dfit($2 \times$ n $\times$ (n $-$ 1)) – double array**

Auxiliary outputs.

If **typ** = 'T', the first $n(n-1)/2$ elements contain the distances, $\hat{d}_{ij}$, for the points returned in **x**, the second set of $n(n-1)/2$ contains the distances $\hat{d}_{ij}$ ordered by the input distances, d_{ij}, the third set of $n(n-1)/2$ elements contains the monotonic distances, $\tilde{d}_{ij}$, ordered by the input distances, d_{ij} and the final set of $n(n-1)/2$ elements contains fitted monotonic distances, $\tilde{d}_{ij}$, for the points in **x**. The $\tilde{d}_{ij}$ corresponding to distances which are input as missing are set to zero.

If **typ** = 'S', the results are as above except that the squared distances are returned.

Each distance matrix is stored in lower triangular packed form in the same way as the input matrix D.

4: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **ndim** < 1,
or **n** $\leq$ **ndim**,
or **typ** $\neq$ 'T' or 'S',
or **ldx** $<$ **n**.

ifail $= 2$

On entry, all elements of **d** ≤ 0.0.

ifail $= 3$

The optimization has failed to converge in **iter** function iterations. Try either increasing the number of iterations using **iter** or increasing the value of ϵ, given by **iopt**, used to determine convergence. Alternatively try a different starting configuration.

ifail $= 4$

The conditions for an acceptable solution have not been met but a lower point could not be found. Try using a larger value of ϵ, given by **iopt**.

ifail $= 5$

The optimization cannot begin from the initial configuration. Try a different set of points.

ifail $= 6$

The optimization has failed. This error is only likely if **iopt** < 0. It corresponds to **ifail** $= 4$, 7 and 9 in e04dg.

7 Accuracy

After a successful optimization the relative accuracy of **stress** should be approximately ϵ, as specified by **iopt**.

8 Further Comments

The optimization function e04dg used by g03fc has a number of options to control the process. The options for the maximum number of iterations (**Iteration Limit**) and accuracy (**Optimality Tolerance**) can be controlled by **iter** and **iopt** respectively. The printing option (**Print Level**) is set to -1 to give no printing. The other option set is to stop the checking of derivatives (**Verify** $=$ No) for efficiency. All other options are left at their default values. If however **iopt** < 0 is used, only the maximum number of iterations is set. All other options can be controlled by the option setting mechanism of e04dg with the defaults as given by that function.

Missing values in the input distance matrix can be specified by a negative value and providing there are not more than about two thirds of the values missing the algorithm may still work. However the function g03fa does not allow for missing values so an alternative method of obtaining an initial set of co-ordinates is required. It may be possible to estimate the missing values with some form of average and then use g03fa to give an initial set of co-ordinates.

9 Example

```
typ = 'T';
n = int32(14);
```

```
d = [0.099;
      0.033;
      0.022;
      0.183;
      0.114;
      0.042;
      0.148;
      0.224;
      0.059;
      0.068;
      0.198;
      0.039;
      0.053;
      0.08500000000000001;
      0.051;
      0.462;
      0.266;
      0.322;
      0.435;
      0.268;
      0.025;
      0.628;
      0.442;
      0.444;
      0.406;
      0.24;
      0.129;
      0.014;
      0.113;
      0.07000000000000001;
      0.046;
      0.047;
      0.034;
      0.002;
      0.106;
      0.129;
      0.173;
      0.119;
      0.162;
      0.331;
      0.177;
      0.039;
      0.089;
      0.237;
      0.07099999999999999;
      0.434;
      0.419;
      0.339;
      0.505;
      0.469;
      0.39;
      0.315;
      0.349;
      0.151;
      0.43;
      0.762;
      0.633;
      0.781;
      0.7;
      0.758;
      0.625;
      0.469;
      0.618;
      0.44;
      0.538;
      0.607;
      0.53;
      0.389;
      0.482;
      0.579;
```

```
      0.597;
      0.498;
      0.374;
      0.5620000000000001;
      0.247;
      0.383;
      0.387;
      0.08400000000000001;
      0.586;
      0.435;
      0.55;
      0.53;
      0.552;
      0.509;
      0.369;
      0.471;
      0.234;
      0.346;
      0.456;
      0.09;
      0.038];
x = [0.240788133045093, 0.2336771621940003;
      0.1136560325924034, 0.1167860264892129;
      0.2393598093421404, 0.07600313166778161;
      0.2129341229512031, 0.06047901667594589;
      0.249489547604401, -0.0693317659454353;
      0.1487285493810621, -0.07783569123014553;
      -0.05139397395173806, -0.1623059860926901;
      0.01153621118615053, -0.3446314902007094;
      -0.003932616383228558, 0.00590873285110863;
      0.03856929371547055, -0.008874156731281325;
      -0.04211582109685088, -0.05655546538851101;
      -0.5158303485449264, 0.02909775244965832;
      -0.3180272688082665, 0.1500964500879802;
      -0.3237616710329136, 0.04748628317308473];
iter = int32(0);
iopt = int32(0);
[xOut, stress, dfit, ifail] = g03fc(typ, n, d, x, iter, iopt)
```

```
xOut =
    0.2060    0.2438
    0.1063    0.1418
    0.2224    0.0817
    0.3032    0.0355
    0.2645   -0.0698
    0.1554   -0.0435
   -0.0070   -0.1612
    0.0749   -0.3275
    0.0488    0.0289
    0.0124   -0.0267
   -0.1649   -0.2500
   -0.5073    0.1267
   -0.3093    0.1590
   -0.3498    0.0700
stress =
    0.1256
dfit =
     array elided
ifail =
           0
```