NAG Fortran Library Manual
Mark 19

Volume 10

G03 – G07

G03 – Multivariate Methods
G04 – Analysis of Variance
G05 – Random Number Generators
G07 – Univariate Estimation

NAG Fortran Library Manual, Mark 19

©The Numerical Algorithms Group Limited, 1999

All rights reserved. No part of this manual may be reproduced, transcribed, stored in a retrieval system, translated into any language or computer language or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the copyright owner.

The copyright owner gives no warranties and makes no representations about the contents of this manual and specifically disclaims any implied warranties or merchantability or fitness for any purpose.

The copyright owner reserves the right to revise this manual and to make changes from time to time in its contents without notifying any person of such revisions or changes.

September 1999 ISBN 1-85206-169-3

NAG is a registered trademark of:
The Numerical Algorithms Group Limited
The Numerical Algorithms Group Inc
The Numerical Algorithms Group (Deutschland) GmbH
Nihon Numerical Algorithms Group KK

All other trademarks are acknowledged.

NAG Ltd
Wilkinson House
Jordan Hill Road
Oxford
OX2 8DR
United Kingdom

Tel: +44 (0)1865 511245
Fax: +44 (0)1865 310139

Nihon NAG KK
Nagashima Building 2F
2-24-3 Higashi
Shibuya-ku
Tokyo
Japan

Tel: +81 (0)3 5485 2901
Fax: +81 (0)3 5485 2903

NAG GmbH
Schleißheimerstraße 5
85748 Garching
Deutschland

Tel: +49 (0)89 3207395
Fax: +49 (0)89 3207396

NAG Inc
1400 Opus Place, Suite 200
Downers Grove, IL 60515-5702
USA

Tel: +1 630 971 2337
Fax: +1 630 971 2706

NAG also has a number of distributors throughout the world. Please contact NAG for further details.

[*NP3390/19*]

Chapter G03 – Multivariate Methods

Note. Please refer to the Users' Note for your implementation to check that a routine is available.

Routine Name	Mark of Introduction	Purpose
G03AAF	14	Performs principal component analysis
G03ACF	14	Performs canonical variate analysis
G03ADF	14	Performs canonical correlation analysis
G03BAF	15	Computes orthogonal rotations for loading matrix, generalized orthomax criterion
G03BCF	15	Computes Procrustes rotations
G03CAF	15	Computes the maximum likelihood estimates of the parameters of a factor analysis model, factor loadings, communalities and residual correlations
G03CCF	15	Computes factor score coefficients (for use after G03CAF)
G03DAF	15	Computes test statistic for equality of within-group covariance matrices and matrices for discriminant analysis
G03DBF	15	Computes Mahalanobis squared distances for group or pooled variance-covariance matrices (for use after G03DAF)
G03DCF	15	Allocates observations to groups according to selected rules (for use after G03DAF)
G03EAF	16	Computes distance matrix
G03ECF	16	Hierarchical cluster analysis
G03EFF	16	K-means cluster analysis
G03EHF	16	Constructs dendrogram (for use after G03ECF)
G03EJF	16	Computes cluster indicator variable (for use after G03ECF)
G03FAF	17	Performs principal co-ordinate analysis, classical metric scaling
G03FCF	17	Performs non-metric (ordinal) multidimensional scaling
G03ZAF	15	Produces standardized values (z-scores) for a data matrix

Chapter G03

Multivariate Methods

Contents

1 Scope of the Chapter

This chapter is concerned with methods for studying multivariate data. A multivariate data set consists of several variables recorded on a number of objects or individuals. Multivariate methods can be classified as those that seek to examine the relationships between the variables (e.g. principal components), known as variable-directed methods, and those that seek to examine the relationships between the objects (e.g. cluster analysis), known as individual-directed methods.

Multiple regression is not included in this chapter as it involves the relationship of a single variable, known as the response variable, to the other variables in the data set, the explanatory variables. Routines for multiple regression are provided in Chapter G02.

2 Background to the Problems

2.1 Variable-directed Methods

Let the n by p data matrix consist of p variables, $x_1, x_2, \ldots, x_p$, observed on n objects or individuals. Variable-directed methods seek to examine the linear relationships between the p variables with the aim of reducing the dimensionality of the problem. There are different methods depending on the structure of the problem. **Principal component analysis** and **factor analysis** examine the relationships between all the variables. If the individuals are classified into groups then **canonical variate analysis** examines the between group structure. If the variables can be considered as coming from two sets then **canonical correlation analysis** examines the relationships between the two sets of variables. All four methods are based on an eigenvalue decomposition or a singular value decomposition (SVD) of an appropriate matrix.

The above methods may reduce the dimensionality of the data from the original p variables to a smaller number, k, of derived variables that adequately represent the data. In general these k derived variables will be unique only up to an **orthogonal rotation**. Therefore it may be useful to see if there exists suitable rotations of these variables that lead to a simple interpretation of the new variables in terms of the original variables.

2.1.1 Principal component analysis

Principal component analysis finds new variables which are linear combinations of the p observed variables so that they have maximum variation and are orthogonal (uncorrelated).

Let S be the p by p variance-covariance matrix of the n by p data matrix. A vector a_1 of length p is found such that:

$$a_1^T S a_1 \text{ is maximised subject to } a_1^T a_1 = 1.$$

The variable $z_1 = \sum_{i=1}^{p} a_{1i} x_i$ is known as the first principal component and gives the linear combination of the variables that gives the maximum variation. A second principal component, $z_2 = \sum_{i=1}^{p} a_{2i} x_i$, is found such that:

$$a_2^T S a_2 \text{ is maximised subject to } a_2^T a_2 = 1 \text{ and } a_2^T a_1 = 0.$$

This gives the linear combination of variables, orthogonal to the first principal component, that gives the maximum variation. Further principal components are derived in a similar way.

The vectors a_i, for $i = 1, 2, \ldots, p$ are the eigenvectors of the matrix S and associated with each eigenvector is the eigenvalue, γ_i^2. The value of $\gamma_i^2 / \sum \gamma_i^2$ gives the proportion of variation explained by the ith principal component. Alternatively the a_i can be considered as the right singular vectors in a SVD of a scaled mean centred data matrix. The singular values of the SVD are the γ_i-values.

Often fewer than p dimensions (principal components) are needed to represent most of the variation in the data. A test on the smaller eigenvalues can be used to investigate the number of dimensions needed.

The values of the principal component variables for the individuals are known as the principal component scores. These can be standardized so that the variance of these scores for each principal component is 1.0 or equal to the corresponding eigenvalue. The principal component scores correspond to the left-hand singular vectors in the SVD.

2.1.2 Factor analysis

Let the p variables have variance-covariance matrix Σ. The aim of factor analysis is to account for the covariances in these p variables in terms of a smaller number, k, of hypothetical variables, or factors, $f_1, f_2, \ldots, f_k$. These are assumed to be independent and to have unit variance. The relationship between the observed variables and the factors is given by the model

$$x_i = \sum_{j=1}^{k} \lambda_{ij} f_j + e_i \quad i = 1, 2, \ldots, p$$

where λ_{ij}, for $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, k$, are the factor loadings and e_i, for $i = 1, 2, \ldots, p$, are independent random variables with variances ψ_i. These represent the unique component of the variation of each observed variable. The proportion of variation for each variable accounted for by the factors is known as the communality.

The model for the variance-covariance matrix, Σ, can then be written as:

$$\Sigma = \Lambda \Lambda^T + \Psi,$$

where Λ is the matrix of the factor loadings, λ_{ij}, and Ψ is a diagonal matrix of the unique variances ψ_i.

If it is assumed that both the k factors and the e_i follow independent Normal distributions then the parameters of the model, Λ and Ψ, can be estimated by maximum likelihood as described by Lawley and Maxwell [7]. The computation of the maximum likelihood estimates is an iterative procedure which involves computing the eigenvalues and eigenvectors of the matrix

$$S^* = \Psi^{-1/2} S \Psi^{-1/2},$$

where S is the sample variance-covariance matrix. Alternatively the SVD of the matrix $R\Psi^{-1/2}$ can be used, where $R^T R = S$. When convergence has been achieved the estimates $\hat{\Lambda}$, of Λ, are obtained by scaling the eigenvectors of S^*. The use of maximum likelihood estimation means that likelihood ratio tests can be constructed to test for the number of factors required.

Having found the estimates of the parameters of the model, the estimates of the values of the factors for the individuals, the **factorscores**, can be computed. These involve the calculation of the **factor score coefficients**. Two common methods of computing factor score coefficients are the regression method and Bartlett's method. Bartlett's method gives unbiased estimates of the factor scores while estimates from the regression method are biased but have smaller variance than those from Bartlett's method; see Lawley and Maxwell [7].

2.1.3 Canonical variate analysis

If the individuals can be classified into one of g groups then canonical variate analysis finds the linear combinations of the p variables that maximize the ratio of the between group variation to the within-group variation. These variables are known as canonical variates. As the canonical variates provide discrimination between the groups the method is also known as **canonical discrimination**.

The canonical variates can be calculated from the eigenvectors of the within group sums of squares and cross-products matrix or from the SVD of the matrix

$$V = Q_x^T Q_g,$$

where Q_g is an orthogonal matrix that defines the groups and Q_x is the first p columns of the orthogonal matrix Q from the QR decompostion of the data matrix with the variable means subtracted. If the data matrix is not of full rank the Q_x matrix can be obtained from a SVD. If the SVD of V is

$$V = U_x \Delta U_g^T,$$

then the non-zero elements ($\delta_i > 0$) of the diagonal matrix Δ are the canonical correlations. The largest δ_i is called the **first canonical correlation** and associated with it is the first canonical variate.

The eigenvalues, γ_i^2, of the within-group sums of squares matrix are given by:

$$\gamma_i^2 = \frac{\delta_i^2}{1 - \delta_i^2}$$

and the value of $\pi_i = \gamma_i^2 / \sum \gamma_i^2$ gives the proportion of variation explained by the ith canonical variate. The values of the π_i give an indication as to how many canonical variates are needed to adequately describe the data, i.e., the dimensionality of the problem. The number of dimensions can be investigated by means of a test on the smaller canonical correlations.

The canonical variate loadings and the relationship between the original variables and the canonical variates are calculated from the matrix U_x. This matrix is scaled so that the canonical variates have unit variance.

2.1.4 Canonical correlation analysis

If the p variables can be considered as coming from two sets then canonical correlation analysis finds linear combinations of the variables in each set, known as canonical variates, such that the correlations between corresponding canonical variates for the two sets are maximized. Let the two sets of variables be denoted by x and y with p_x and p_y variables in each set respectively. Let the variance-covariance of the data set be

$$S = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{yx} & S_{yy} \end{bmatrix}$$

and let

$$\Sigma = S_{yy}^{-1} S_{yx} S_{xx}^{-1} S_{xy}$$

then the canonical correlations can be calculated from the eigenvalues of the matrix Σ. Alternatively the canonical correlations can be calculated by means of a SVD of the matrix

$$V = Q_x^T Q_y,$$

where Q_x is the first p_x columns of the orthogonal matrix Q from the QR decompostion of the x-variables in the data matrix and Q_y is the first p_y columns of the Q matrix of the QR decomposition of the y-variables in the data matrix. In both cases the variable means are subtracted before the QR decomposition is computed. If either sets of variables is not of full rank an SVD can be used instead of the QR decomposition. If the SVD of V is

$$V = U_x \Delta U_y^T,$$

then the non-zero elements ($\delta_i > 0$) of the diagonal matrix Δ are the canonical correlations. The largest δ_i is called the **first canonical correlation** and associated with it is the first canonical variate. The eigenvalues, γ_i^2, of the matrix Σ are given by

$$\gamma_i^2 = \frac{\delta_i^2}{1 + \delta_i^2}.$$

The value of $\pi_i = \gamma_i^2 / \sum \gamma_i^2$ gives the proportion of variation explained by the ith canonical variate. The values of the π_i give an indication as to how many canonical variates are needed to adequately describe the data, i.e., the dimensionality of the problem; this can also be investigated by means of a test on the smaller values of the γ_i^2.

The relationship between the canonical variables and the original variables, the canonical variate loadings, can be computed from the U_x and U_y matrices.

2.1.5 Rotations

There are two principal reasons for using rotations. Either

(a) simplifying the structure to aid interpretation of derived variables, or
(b) comparing two or more data sets or sets of derived variables.

The most common type of rotations used for (a) are **orthogonal rotations**. If Λ is the p by k loading matrix from a variable-directed multivariate method, then the rotations are selected such that the elements, λ_{ij}^*, of the rotated loading matrix, Λ^*, are either relatively large or small. The rotations may be found by minimizing the criterion

$$V = \sum_{j=1}^{k} \sum_{i=1}^{p} (\lambda_{ij}^*)^4 - \frac{\gamma}{p} \sum_{j=1}^{k} \left(\sum_{i=1}^{p} (\lambda_{ij}^*)^2 \right)^2$$

where the constant, γ, gives a family of rotations, with $\gamma = 1$ giving **varimax rotations** and $\gamma = 0$ giving **quartimax** rotations.

For (b) **Procrustes** rotations are used. Let A and B be two l by m matrices, which can be considered as representing l points in m dimensions. One example is if A is the loading matrix from a variable-directed multivariate method and B is a hypothesised pattern matrix. In order to try to match the points in A and B there are three steps:

(i) translate so that centroids of both matrices are at the origin,
(ii) find a rotation that minimizes the sum of squared distances between corresponding points of the matrices,
(iii) scale the matrices.

For a more detailed description, see Krzanowski [6].

2.2 Individual-directed Methods

While dealing with the same n by p data matrix as variable-directed methods the emphasis is the n objects or individuals rather than the p variables. The methods are generally based on an n by n distance or dissimilarity matrix such that the (k, j)th element gives a measure of how 'far apart' individual k and j are. Alternatively, a similarity matrix can be used which measures how 'close' individuals are. The form of the measure of distance or similarity will depend upon the form of the p variables. For continuous variables it is usually assumed that some form of Euclidean distance is suitable. That is, for x_{ki} and x_{ji} measured for individuals k and j on variable i respectively, the contribution to distance between individuals k and j from variable i is given by

$$(x_{ki} - x_{ji})^2.$$

Often there will be a need to scale the variables to produce satisfactory distances. For discrete variables there are various measures of similarity or distance that can easily be computed. For example, for binary data a measure of similarity could be

1 – if the individuals take the same value,

0 – otherwise.

Given a measure of distance between individuals there are three basic tasks that can be performed.

(1) *Group the individuals;* that is, collect the individuals into groups so that those within a group are closer to each other than they are to members of another group.
(2) *Classify individuals;* that is, if some individuals are known to come from certain groups allocate individuals whose group membership is unknown to the nearest group.
(3) *Map the individuals;* that is, produce a multidimensional diagram in which the distances on the diagram represent the distances between the individuals.

In the above, (1) leads to cluster analysis, (2) leads to discriminant analysis and (3) leads to scaling methods.

2.2.1 Hierarchical cluster analysis

Approaches for cluster analysis can be classified into two types: hierarchical and non-hierarchical. Hierarchical cluster analysis produces a series of overlapping groups or clusters ranging from separate individuals to one single cluster. For example five individuals could be hierarchically clustered as follows.

Step 1	(1)	(2)	(3)	(4)	(5)
Step 2	(1,2)		(3,4)		(5)
Step 3	(1,2)		(3,4,5)		
Step 4	(1,2,3,4,5)				

The clusters at a level are constructed from the clusters at a previous level. There are two basic approaches to hierarchical cluster analysis: agglomerative methods which build up clusters starting from individuals until there is only one cluster, or divisive methods which start with a single cluster and split clusters until the individual level is reached. This chapter contains the more common agglomerative methods.

The stages in a hierarchical cluster analysis are usually as follows.

(a) Form a distance matrix
(b) Use selected criterion to form hierarchy.
(c) Print cluster information in the form of a dendrogram or use information to form a set of clusters.

These three stages will be considered in turn.

(a) Form distance matrix.

For the n by p data matrix X, a general measure of the distance between object j and object k, d_{jk}, is:

$$d_{jk} = \left(\sum_{i=1}^{p} D(x_{ji}/s_i, x_{ki}/s_i)\right)^{\alpha},$$

where x_{ji} and x_{ki} are the (j,i)th and (k,i)th elements of X, s_i is a standardization for the ith variable and $D(u,v)$ is a suitable function. Three common distances for continuous variables are:

(i) Euclidean distance: $D(u,v) = (u-v)^2$ and $\alpha = \frac{1}{2}$.

(ii) Euclidean squared distance: $D(u,v) = (u-v)^2$ and $\alpha = 1$.

(iii) Absolute distance (city block metric): $D(u,v) = |u-v|$ and $\alpha = 1$.

The common standardisations are the standard deviation and the range. For dichotomous variables there are a number of different measures (see Krzanowski [6] and Everitt [2]); these are usually easy to compute. If the individuals in a cluster analysis are themselves variables, then a suitable distance measure will be based on the correlation coefficient for continuous variables and contingency table statistics for discrete data.

(b) Form Hierarchy

Given a distance matrix for the n individuals, an agglomerative clustering methods produces a hierarchical tree by starting with n clusters each with a single individual and then at each of $n-1$ stages merging two clusters to form a larger cluster until all individuals are in a single cluster. At each stage the two clusters that are nearest are merged to form a new cluster and a new distance matrix is computed for the reduced number of clusters.

Methods differ as to how the distances between the new cluster and other clusters are computed. For three clusters i, j and k let n_i, n_j and n_k be the number of objects in each cluster and let d_{ij}, d_{ik} and d_{jk} be the distances between the clusters. If clusters j and k be merged to give cluster jk, then the distance from cluster i to cluster jk, $d_{i.jk}$, can be computed in the following ways.

(a) Single Link or nearest neighbour : $d_{i.jk} = \min(d_{ij}, d_{ik})$.

(b) Complete Link or furthest neighbour : $d_{i.jk} = \max(d_{ij}, d_{ik})$.

(c) Group average : $d_{i.jk} = \frac{n_j}{n_j+n_k} d_{ij} + \frac{n_k}{n_j+n_k} d_{ik}$.

(d) Centroid : $d_{i.jk} = \frac{n_j}{n_j+n_k} d_{ij} + \frac{n_k}{n_j+n_k} d_{ik} - \frac{n_j n_k}{(n_j+n_k)^2} d_{jk}$.

(e) Median : $d_{i.jk} = \frac{1}{2} d_{ij} + \frac{1}{2} d_{ik} - \frac{1}{4} d_{jk}$.

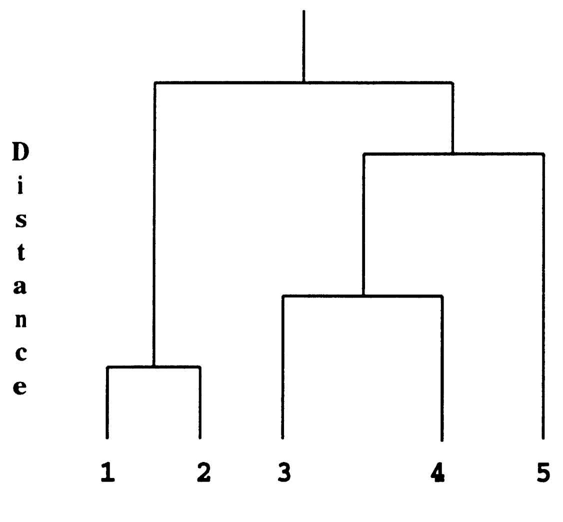
(f) Minimum variance : $d_{i.jk} = [(n_i+n_j)d_{ij} + (n_i+n_k)d_{ik} - n_i d_{jk}]/(n_i+n_j+n_k)$

For further details, see Everitt [2] or Krzanowski [6].

(c) Produce Dendrogram and Clusters

Hierarchical cluster analysis can be represented by a tree that shows at which distance the clusters merge. Such a tree is known as a dendrogram; see Everitt [2] and Krzanowski [6].

A simple example is



Individuals

Figure 1

The end-points of the dendrogram represent the individuals that have been clustered.

Alternatively the information from the tree can be used to produced either a chosen number of clusters or the clusters that exist at a given distance. The latter is equivalent to taking the dendrogram and drawing a line across at a given distance to produce clusters.

2.2.2 Non-hierarchical clustering

Non-heirarchical cluster analysis usually forms a given number of clusters from the data. There is no requirement that if first $k-1$ and then k clusters were requested then the $k-1$ clusters would be formed from the k clusters.

Most non-hierarchical methods of cluster analysis seek to partition the set of individuals into a number of clusters so as to optimise a criterion. The number of clusters is usually specified prior to the analysis. One commonly used criterion is the within-cluster sum of squares. Given n individuals with p variables measured on each individual, x_{ij} for $i = 1,2,\ldots, n$, $j = 1,2,\ldots p$, the within-cluster sum of squares for K clusters is:

$$SS_c = \sum_{k=1}^{K} \sum_{i\in S_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2,$$

where S_k is the set of objects in the kth cluster and $\bar{x}_{kj}$ is the mean for the variable j over cluster k. Starting with an initial allocation of individuals to clusters the method then seeks to minimise SS_c by a series of re-allocations. This is often known as K-means clustering.

2.2.3 Discriminant analysis

Discriminant analysis is concerned with the **allocation** of objects to n_g groups on the basis of observations on those objects using an allocation rule. This rule is computed from observations coming from a **training** set in which group membership is known. The allocation rule is based on the distance between the object and an estimate of the location of the groups. If p variables are observed and the vector of means for the jth group in the training set are $\bar{x}_j$ then the usual measure of the distance of an observation, x_k, from the jth group mean is given by Mahalanobis distance:

$$D_{kj}^2 = (x_k - \bar{x}_j)^T S_*^{-1} (x_k - \bar{x}_j),$$

where S_* is either the within-group variance-covariance matrix, S_j, for the n_j objects in the jth group, or a pooled variance-covariance matrix, S, computed from all n objects from all groups where

$$S = \frac{\sum_{j=1}^{n_g} (n_j - 1) S_j}{(n - n_g)}.$$

If the within group variance-covariance matrices can be assumed to be equal then the pooled variance-covariance matrix can be used. This assumption can be tested using the test statistic:

$$G = C\left((n-n_g)\log|S| - \sum_{j=1}^{n_g}(n_j-1)\log|S_j|\right),$$

where

$$C = 1 - \frac{2p^2+3p-1}{6(p+1)(n_g-1)}\left(\sum_{j=1}^{n_g}\frac{1}{(n_j-1)} - \frac{1}{(n-n_g)}\right).$$

For large n, G is approximately distributed as a χ^2 variable with $\frac{1}{2}p(p+1)(n_g-1)$ degrees of freedom; see Morrison [8].

In addition to the distances a set of prior probabilities of group membership, π_j, for $j = 1, 2, \ldots, n_g$, may be used. The prior probabilities reflect the user's view as to the likelihood of the objects coming from the different groups.

It is generally assumed that the p variables follow a multivariate Normal distribution with, for the jth group, mean μ_j and variance-covariance matrix Σ_j. If $p(x_k|\mu_j,\Sigma_j)$ is the probability of observing the observation x_k from group j, then the posterior probability of belonging to group j is

$$p(j|x_k,\mu_j,\Sigma_j) \propto p(x_k|\mu_j,\Sigma_j)\pi_j.$$

An observation is allocated to the group with the highest posterior probability.

In the **estimative** approach to discrimination the parameters μ_j and Σ_j in $p(j|x_k,\mu_j,\Sigma_j)$ are replaced by their estimates calculated from the training set. If it is assumed that the within-group variance-covariance matrices are equal then the **linear discriminant function** is obtained; otherwise if it is assumed that the variance-covariance matrices are unequal then the **quadratic discriminant function** is obtained.

In the Bayesian **predictive** approach a non-informative prior distribution is used for the parameters giving the posterior distribution for the parameters from the training set, X_t, of, $p(\mu_j,\Sigma_j|X_t)$. A predictive distribution is then obtained by integrating $p(j|x_k,\mu_j,\Sigma_j)p(\mu_j,\Sigma_j|X)$ over the parameter space. This predictive distribution, $p(x_k|X_t)$, then replaces $p(x_k|\mu_j,\Sigma_j)$ to give

$$p(j|x_k,\mu_j,\Sigma_j) \propto p(x_k|X_t)\pi_j.$$

In addition to allocating the objects to groups an atypicality index for each object and for each group can be computed. This represents the probability of obtaining an observation more typical of the group than that observed. A high value of the atypicality index for all groups indicates that the observation may in fact come from a group not represented in the training set.

Alternative approaches to discrimination are the use of canonical variates and logistic discrimination. Canonical variate analysis is described above and as it seeks to find the directions that best discriminate between groups these directions can also be used to allocate further observations. This can be viewed as an extension of **Fisher's linear discriminant function**. This approach does not assume that the data is Normally distributed, but Fisher's linear discriminant function may not perform well on non-Normal data. In the case of two groups, logistic regression can be performed with the response variable indicating the group allocation and the variables in the discriminant analysis being the explanatory variables. Allocation can then be made on the basis of the fitted response value. This is known as **logistic discrimination** and can be shown to be valid for a wide range of distributional assumptions.

2.2.4 Scaling methods

Scaling methods seek to represent the observed dissimilarities or distances between objects as distances between points in Euclidean space. For example if the distances between objects A, B and C were 3, 4 and 5 the distances could be represented exactly by three points in two-dimensional space. Only their relative positions would be important, the whole configuration of points could be rotated or shifted without effecting the distances between the points. If a one-dimensional representation was required the 'best' representation might give distances of $2\frac{1}{3}, 3\frac{1}{3}$ and $5\frac{2}{3}$, which may be an adequate representation. If the distances were 3, 4 and 8 then these distances could not be exactly represented in Euclidean space even in two dimensions; the best representation being the three points in a straight line giving distances 3, 4 and 7.

In practice the user of scaling methods has to decide upon the number of dimensions in which the data is to be represented. The smaller the number the easier it will be to assimilate the information. The chosen number of dimensions needs to give an adequate representation of the data but will often not give an exact representation because either the number of chosen dimensions is too small or the data cannot be represented in Euclidean space.

Two basic methods are available depending on the nature of the dissimilarities or distances being analysed. If the distances can be assumed to satisfy the metric inequality

$$d_{ij} \leq d_{ik} + d_{kj},$$

then the distances can be represented exactly by points in Euclidean space and the technique known as metric scaling, classical scaling or principal coordinate analysis can be used. This technique involves the computing of the eigenvalues of a matrix derived from the distance matrix. The eigenvectors corresponding to the k largest positive eigenvalues gives the best k dimensions in which to represent the objects. If there are negative eigenvalues then the distance matrix cannot be represented in Euclidean space.

Instead of the above approach of requiring the distances from the points to match the distances from the objects as closely as possible sometimes only a rank-order equivalence is required. That is, the ith largest distance between objects should, as far as possible, be represented by the ith largest distance between points. This would be appropriate when the dissimilarities are based on subjective rankings. For example if the objects were foods the a number of judges rank the foods for different qualities such as taste and texture the resulting distances would not necessarily obey the metric inequality but the rank order would be significant. Alternatively, by relaxing the requirement from matching distances to rank order equivalence only, the number of dimensions required to represent the distance matrix may be decreased. The requirement of rank-order equivalence leads to non-metric or ordinal multidimensional scaling. The criterion used to measure the closeness of the fitted distance matrix to the observed distance matrix is known as STRESS which is given by

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

where $\hat{d}_{ij}^2$ is the Euclidean squared distance between the computed points i and j and $\tilde{d}_{ij}$ is the fitted distance obtained when $\hat{d}_{ij}$ is monotonically regressed on the observed distances 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. Thus STRESS is a measure of by how much the set of points preserve the order of the distances in the original distance matrix and non-metric multidimensional scaling seeks to find the set of points that minimize the STRESS.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

The following routines perform the computations for variable-directed methods.

G03AAF	computes the principal components from an input data matrix. Results include tests on the eigenvalues, the principal component loadings, and the principal component scores.
G03ACF	computes a canonical variate analysis from an input data matrix. Results include canonical correlations, tests on eigenvalues, canonical variate means, and canonical variate loadings.
G03ADF	computes a canonical correlation analysis from a input data matrix. Results include tests on the eigenvalues and canonical variates loadings.
G03CAF	computes maximum likelihood estimates of the parameters of the factor analysis model.
G03CCF	computes the factor score coefficients from the results of G03CAF.
G03BAF	computes orthogonal rotations, including varimax and equimax rotations.
G03BCF	computes Procrustes rotations.

The following routines perform the computations for individual-directed methods.

Discriminant Analysis

G03DAF	computes matrices for use in discriminant analysis and test statistics for use in testing the equality of within group variance-covariance matrices.
G03DBF	computes Mahalanobis distances from the results of G03DAF.
G03DCF	allocates observations to groups using allocation rules as described above. An atypicality index can also be computed. G03DCF uses the results of G03DAF.

Note also that G02GBF will fit a logistic regression model and can be used for logistic discrimination.

Cluster Analysis

G03EAF	computes a distance matrix.
G03ECF	computes heirarchical cluster analysis from a given distance matrix.
G03EHF	computes a dendrogram from the results of G03ECF.
G03EJF	computes a set of clusters from the results of G03ECF.
G03EFF	computes non-heirarchical (K-means) cluster analysis.

Scaling Methods

G03FAF	computes a principal co-ordinate analysis.
G03FCF	computes non-metric multi-dimensional scaling.

The following service routine is also available:

G03ZAF	computes a matrix of standardized variables from an input data matrix.

4 References

[1] Chatfield C and Collins A J (1980) *Introduction to Multivariate Analysis* Chapman and Hall

[2] Everitt B S (1974) *Cluster Analysis* Heinemann

[3] Gnanadesikan R (1977) *Methods for Statistical Data Analysis of Multivariate Observations* Wiley

[4] Hammarling S (1985) The singular value decomposition in multivariate statistics *SIGNUM Newsl.* **20** **(3)** 2–25

[5] Kendall M G and Stuart A (1976) *The Advanced Theory of Statistics (Volume 3)* Griffin (3rd Edition)

[6] Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

[7] Lawley D N and Maxwell A E (1971) *Factor Analysis as a Statistical Method* Butterworths (2nd Edition)

[8] Morrison D F (1967) *Multivariate Statistical Methods* McGraw-Hill

G03AAF – NAG Fortran Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G03AAF performs a principal component analysis on a data matrix; both the principal component loadings and the principal component scores are returned.

2 Specification

```
SUBROUTINE G03AAF(MATRIX, STD, WEIGHT, N, M, X, LDX, ISX, S, WT,
1                 NVAR, E, LDE, P, LDP, V, LDV, WK, IFAIL)
 INTEGER          N, M, LDX, ISX(M), NVAR, LDE, LDP, LDV, IFAIL
 real             X(LDX,M), S(M), WT(*), E(LDE,6), P(LDP,NVAR),
1                 V(LDV,NVAR), WK(NVAR*NVAR+5*(NVAR-1))
 CHARACTER*1      MATRIX, STD, WEIGHT
```

3 Description

Let X be an n by p data matrix of n observations on p variables $x_1, x_2, \ldots, x_p$ and let the p by p variance-covariance matrix of $x_1, x_2, \ldots, x_p$ be S. A vector a_1 of length p is found such that:

$a_1^T S a_1$ is maximized subject to $a_1^T a_1 = 1$.

The variable $z_1 = \sum_{i=1}^{p} a_{1i} x_i$ is known as the first principal component and gives the linear combination of the variables that gives the maximum variation. A second principal component, $z_2 = \sum_{i=1}^{p} a_{2i} x_i$, is found such that:

$a_2^T S a_2$ is maximized subject to $a_2^T a_2 = 1$ and $a_2^T a_1 = 0$.

This gives the linear combination of variables that is orthogonal to the first principal component that gives the maximum variation. Further principal components are derived in a similar way.

The vectors $a_1, a_2, \ldots, a_p$, are the eigenvectors of the matrix S and associated with each eigenvector is the eigenvalue, λ_i^2. The value of $\lambda_i^2 / \sum \lambda_i^2$ gives the proportion of variation explained by the ith principal component. Alternatively the a_i's can be considered as the right singular vectors in a singular value decomposition with singular values λ_i of the data matrix centred about its mean and scaled by $1/\sqrt{(n-1)}$, X_s. This latter approach is used in G03AAF, with

$$X_s = V \Lambda P'$$

where Λ is a diagonal matrix with elements λ_i, P' is the p by p marix with columns a_i and V is an n by p matrix with $V'V = I$, which gives the principal component scores.

Principal component analysis is often used to reduce the dimension of a data set, replacing a large number of correlated variables with a smaller number of orthogonal variables that still contain most of the information in the original data set.

The choice of the number of dimensions required is usually based on the amount of variation accounted for by the leading principal components. If k principal components are selected then a test of the equality of the remaining $p-k$ eigenvalues is

$$(n - (2p+5)/6) \left\{ - \sum_{i=k+1}^{p} \log(\lambda_i^2) + (p-k) \log \left(\sum_{i=k+1}^{p} \lambda_i^2 / (p-k) \right) \right\}$$

which has, asymptotically, a χ^2 distribution with $\frac{1}{2}(p-k-1)(p-k+2)$ degrees of freedom.

Equality of the remaining eigenvalues indicates that if any more principal components are to be considered then they all should be considered.

Instead of the variance-covariance matrix the correlation matrix, the sums of squares and cross-products matrix or a standardised sums of squares and cross-products matrix may be used. In the last case S is replaced by $\sigma^{-\frac{1}{2}}S\sigma^{-\frac{1}{2}}$ for a diagonal matrix σ with positive elements. If the correlation matrix is used the χ^2 approximation for the statistic given above is not valid.

The principal component scores, F, are the values of the principal component variables for the observations. These can be standardised so that the variance of these scores for each principal component is 1.0 or equal to the corresponding eigenvalue.

Weights can be used with the analysis, in which case the matrix X is first centred about the weighted means then each row is scaled by an amount $\sqrt{w_i}$, where w_i is the weight for the ith observation.

4 References

[1] Chatfield C and Collins A J (1980) *Introduction to multivariate analysis.* Chapman and Hall

[2] Cooley W C and Lohnes P R (1971) *Multivariate data analysis.* Wiley

[3] Hammarling S (1985) The Singular Value Decomposition in Multivariate Statistics *Signum Newsl.*. **20 (3)** 2–25

[4] Kendall M G and Stuart A (1979) *The advanced theory of statistics (3 volumes).* Griffin (4th Edition)

[5] Morrison D F (1967) *Multivariate statistical methods.* McGraw-Hill

5 Parameters

1: MATRIX — CHARACTER*1 *Input*

On entry: indicates for which type of matrix the principal component analysis is to be carried out.

If MATRIX = 'C', then it is for the correlation matrix.

If MATRIX = 'S', then it is for a standardised matrix, with standardisations given by S.

If MATRIX = 'U', then it is for the sums of squares and cross-products matrix.

If MATRIX = 'V', then it is for the variance-covariance matrix.

Constraint: MATRIX = 'C', 'S', 'U' or 'V'.

2: STD — CHARACTER*1 *Input*

On entry: indicates if the principal component scores are to be standardised.

If STD = 'S', then the principal component scores are standardised so that $F'F = I$, i.e., $F = X_sP\Lambda^{-1} = V$..

If STD = 'U', then the principal component scores are unstandardised, i.e., $F = X_sP = V\Lambda$.

If STD = 'Z', then the principal component scores are standardised so that they have unit variance.

If STD = 'E', then the principal component scores are standardised so that they have variance equal to the corresponding eigenvalue.

Constraint: STD = 'E', 'S', 'U' or 'Z'.

3: WEIGHT — CHARACTER*1 *Input*

On entry: indicates if weights are to be used.

If WEIGHT = 'U' (Unweighted), then no weights are used.
If WEIGHT = 'W' (Weighted), then weights are used and must be supplied in WT.

Constraint: WEIGHT = 'U' or 'W'.

4: N — INTEGER *Input*

On entry: the number of observations, n.

Constraint: N $\geq$ 2.

5: M — INTEGER *Input*

On entry: the number of variables in the data matrix, m.

Constraint: M $\geq$ 1.

6: X(LDX,M) — ***real*** array *Input*

On entry: X(i,j) must contain the ith observation for the jth variable, for $i = 1,2,\ldots,n$; $j = 1,2,\ldots,m$.

7: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03AAF is called.

Constraint: LDX $\geq$ N.

8: ISX(M) — INTEGER array *Input*

On entry: ISX(j) indicates whether or not the jth variable is to be included in the analysis.

If ISX$(j) > 0$, then the variable contained in the jth column of X is included in the principal component analysis, for $j = 1,2,\ldots,m$.

Constraint: ISX$(j) > 0$ for NVAR values of j.

9: S(M) — ***real*** array *Input/Output*

On entry: the standardisations to be used, if any.

If MATRIX = 'S', then the first m elements of S must contain the standardisation coefficients, the diagonal elements of σ.

Constraint: if ISX$(j) > 0$, then S$(j) > 0.0$, for $j = 1,2,\ldots,m$.

On exit: if MATRIX = 'S', then S is unchanged on exit.

If MATRIX = 'C', then S contains the variances of the selected variables. S(j) contains the variance of the variable in the jth column of X if ISX$(j) > 0$.

If MATRIX = 'U' or 'V', then S is not referenced.

10: WT($*$) — ***real*** array *Input*

On entry: if WEIGHT = 'W', then the first n elements of WT must contain the weights to be used in the principal component analysis.

If WT$(i) = 0.0$, then the ith observation is not included in the analysis. The effective number of observations is the sum of the weights.

If WEIGHT = 'U', then WT is not referenced and the effective number of observations is n.

Constraint: WT$(i) \geq 0.0$, for $i = 1,2,\ldots,n$ and the sum of weights $\geq$ NVAR + 1.

11: NVAR — INTEGER *Input*

On entry: the number of variables in the principal component analysis, p.

Constraint: $1 \leq$ NVAR $\leq$ min(N−1,M}).

12: E(LDE,6) — ***real*** array *Output*

On exit: the statistics of the principal component analysis.

E(i,1), the eigenvalues associated with the ith principal component, λ_i^2, for $i = 1,2,\ldots,p$.

E(i,2), the proportion of variation explained by the ith principal component, for $i = 1,2,\ldots,p$.

E(i,3), the cumulative proportion of variation explained by the first ith principal components, for $i = 1,2,\ldots,p$.

E(i,4), the χ^2 statistics, for $i = 1,2,\ldots,p$.

E(i,5), the degrees of freedom for the χ^2 statistics, for $i = 1,2,\ldots,p$.

If MATRIX $\neq$ 'C', then E(i,6) contains significance level for the χ^2 statistic, for $i = 1,2,\ldots,p$.

If MATRIX = 'C', then E(i,6) is returned as zero.

13: LDE — INTEGER *Input*

On entry: the first dimension of the array E as declared in the (sub)program from which G03AAF is called.

Constraint: LDE $\geq$ NVAR.

14: P(LDP,NVAR) — ***real*** array *Output*

On exit: the first NVAR columns of P contain the principal component loadings, a_i. The jth column of P contains the NVAR coefficients for the jth principal component.

15: LDP — INTEGER *Input*

On entry: the first dimension of the array P as declared in the (sub)program from which G03AAF is called.

Constraint: LDP $\geq$ NVAR.

16: V(LDV,NVAR) — ***real*** array *Output*

On exit: the first NVAR columns of V contain the principal component scores. The jth column of V contains the N scores for the jth principal component.

If WEIGHT = 'W', then any rows for which WT(i) is zero will be set to zero.

17: LDV — INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which G03AAF is called.

Constraint: LDV $\geq$ N.

18: WK(NVAR*NVAR+5*(NVAR−1)) — ***real*** array *Workspace*

19: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, M < 1,
or N < 2,
or NVAR < 1,
or NVAR > M,
or NVAR ≥ N,
or LDX < N,
or LDV < N,
or LDP < NVAR,
or LDE < NVAR,
or MATRIX ≠ 'C', 'S', 'U' or 'V',
or STD ≠ 'S', 'U', 'Z' or 'E',
or WEIGHT ≠ 'U' or 'W'.

IFAIL = 2

On entry, WEIGHT = 'W' and a value of WT < 0.0.

IFAIL = 3

On entry, there are not NVAR values of ISX > 0,
or WEIGHT = 'W' and the effective number of observations is less than NVAR + 1.

IFAIL = 4

On entry, S(j) ≤ 0.0 for some $j = 1, 2, \ldots, m$, when MATRIX = 'S' and ISX(j) > 0.

IFAIL = 5

The singular value decomposition has failed to converge. See F02WEF. This is an unlikley error exit.

IFAIL = 6

All eigenvalues/singular values are zero. This will be caused by all the variables being constant.

7 Accuracy

As G03AAF uses a singular value decomposition of the data matrix, it will be less affected by ill-conditioned problems than traditional methods using the eigenvalue decomposition of the variance-covariance matrix.

8 Further Comments

None.

9 Example

A data set is taken from Cooley and Lohnes [2], it consists of ten observations on three variables. The unweighted principal components based on the variance-covariance matrix are computed and unstandardised principal component scores requested.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03AAF Example Program Text
*     Mark 17 Revised.  NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NMAX, MMAX
      PARAMETER        (NMAX=12,MMAX=3)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, M, N, NVAR
      CHARACTER        MATRIX, STD, WEIGHT
*     .. Local Arrays ..
      real             E(MMAX,6), P(MMAX,MMAX), S(MMAX), V(NMAX,MMAX),
     +                 WK(MMAX*MMAX+5*(MMAX-1)), WT(NMAX), X(NMAX,MMAX)
      INTEGER          ISX(MMAX)
*     .. External Subroutines ..
      EXTERNAL         G03AAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03AAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) MATRIX, STD, WEIGHT, N, M
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         IF (WEIGHT.EQ.'U' .OR. WEIGHT.EQ.'u') THEN
            DO 20 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), WT(I)
   40       CONTINUE
         END IF
         READ (NIN,*) (ISX(J),J=1,M), NVAR
         IF (MATRIX.EQ.'S' .OR. MATRIX.EQ.'s') READ (NIN,*) (S(J),J=1,M)
         IFAIL = 0
*
         CALL G03AAF(MATRIX,STD,WEIGHT,N,M,X,NMAX,ISX,S,WT,NVAR,E,MMAX,
     +               P,MMAX,V,NMAX,WK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +   'Eigenvalues  Percentage  Cumulative    Chisq     DF     Sig'
         WRITE (NOUT,*) '              variation   variation'
         WRITE (NOUT,*)
         DO 60 I = 1, NVAR
            WRITE (NOUT,99999) (E(I,J),J=1,6)
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Eigenvalues'
         WRITE (NOUT,*)
         DO 80 I = 1, NVAR
            WRITE (NOUT,99998) (P(I,J),J=1,NVAR)
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Principal component scores'
         WRITE (NOUT,*)
         DO 100 I = 1, N
            WRITE (NOUT,99997) I, (V(I,J),J=1,NVAR)
  100    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,F11.4,2F12.4,F10.4,F8.1,F8.4)
99998 FORMAT (1X,8F9.4)
99997 FORMAT (1X,I2,(8F9.3))
      END
```

9.2 Example Data

```
G03AAF Example Program Data
 'V' 'E' 'U' 10 3
7.0 4.0 3.0
4.0 1.0 8.0
6.0 3.0 5.0
8.0 6.0 1.0
8.0 5.0 7.0
7.0 2.0 9.0
5.0 3.0 3.0
9.0 5.0 8.0
7.0 4.0 5.0
8.0 2.0 2.0
 1   1   1   3
```

9.3 Example Results

```
 G03AAF Example Program Results

 Eigenvalues  Percentage  Cumulative     Chisq      DF     Sig
               variation   variation

      8.2739      0.6515      0.6515    8.6127     5.0  0.1255
      3.6761      0.2895      0.9410    4.1183     2.0  0.1276
      0.7499      0.0590      1.0000    0.0000     0.0  0.0000

 Eigenvalues

    0.1376   0.6990   0.7017
    0.2505   0.6609  -0.7075
   -0.9583   0.2731  -0.0842

 Principal component scores

  1    2.151   -0.173   -0.107
  2   -3.804   -2.887   -0.510
  3   -0.153   -0.987   -0.269
  4    4.707    1.302   -0.652
  5   -1.294    2.279   -0.449
  6   -4.099    0.144    0.803
  7    1.626   -2.232   -0.803
  8   -2.114    3.251    0.168
  9    0.235    0.373   -0.275
 10    2.746   -1.069    2.094
```

G03ACF – NAG Fortran Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G03ACF performs a canonical variate (canonical discrimination) analysis.

2 Specification

```
      SUBROUTINE G03ACF(WEIGHT, N, M, X, LDX, ISX, NX, ING, NG, WT, NIG,
     1                  CVM, LDCVM, E, LDE, NCV, CVX, LDCVX, TOL,
     2                  IRANKX, WK, IWK, IFAIL)
      INTEGER           N, M, LDX, ISX(M), NX, ING(N), NG, NIG(NG),
     1                  LDCVM, LDE, NCV, LDCVX, IRANKX, IWK, IFAIL
      real              X(LDX,M), WT(*), CVM(LDCVM,NX), E(LDE,6),
     1                  CVX(LDCVX,NG-1), TOL, WK(IWK)
      CHARACTER*1       WEIGHT
```

3 Description

Let a sample of n observations on n_x variables in a data matrix come from n_g groups with $n_1, n_2, \ldots, n_{n_g}$ observations in each group, $\sum n_i = n$. Canonical variate analysis finds the linear combination of the n_x variables that maximizes the ratio of between-group to within-group variation. The variables formed, the canonical variates, can be used to discriminate between groups.

The canonical variates can be calculated from the eigenvectors of the within group sums of squares and cross-products matrix. However, G03ACF calculates the canonical variates by means of a singular value decomposition (SVD) of a matrix V. Let the data matrix with variable (column) means subtracted be X and let its rank be k; then the k by $(n_g - 1)$ matrix V is given by:

$V = Q_X^T Q_g$, where Q_g is an n by $(n_g - 1)$ orthogonal matrix that defines the groups and Q_X is the first k rows of the orthogonal matrix Q either from the QR decompostion of X:

$$X = QR$$

if X is of full column rank, i.e., $k = n_x$, else from the SVD of X:

$$X = QDP^T.$$

Let the SVD of V be:

$$V = U_x \Delta U_g^T$$

then the non-zero elements of the diagonal matrix Δ, δ_i, for $i = 1, 2, \ldots, l$, are the l canonical correlations associated with the l canonical variates, where $l = \min(k, n_g)$.

The eigenvalues, λ_i^2, of the within-group sums of squares matrix are given by:

$$\lambda_i^2 = \frac{\delta_i^2}{1 - \delta_i^2}.$$

and the value of $\pi_i = \lambda_i^2 / \sum \lambda_i^2$ gives the proportion of variation explained by the ith canonical variate. The values of the π_i's give an indication as to how many canonical variates are needed to adequately describe the data, i.e., the dimensionality of the problem.

To test for a significant dimensionality greater than i the χ^2 statistic:

$$(n - 1 - n_g - \frac{1}{2}(k - n_g)) \sum_{j=i+1}^{l} \log(1 + \lambda_j^2)$$

can be used. This is asymptotically distributed as a χ^2 distribution with $(k-i)(n_g-1-i)$ degrees of freedom. If the test for $i=h$ is not significant, then the remaining tests for $i>h$ should be ignored.

The loadings for the canonical variates are calulated from the matrix U_x. This matrix is scaled so that the canonical variates have unit within group variance.

In addition to the canonical variates loadings the means for each canonical variate are calculated for each group.

Weights can be used with the analysis, in which case the weighted means are subtracted from each column and then each row is scaled by an amount $\sqrt{w_i}$, where w_i is the weight for the ith observation (row).

4 References

[1] Chatfield C and Collins A J (1980) *Introduction to multivariate analysis.* Chapman and Hall

[2] Gnanadesikan R (1977) *Methods for statistical data analysis of multivariate observations.* Wiley

[3] Hammarling S (1985) The Singular Value Decomposition in Multivariate Statistics *Signum Newsl.*. **20 (3)** 2–25

[4] Kendall M G and Stuart A (1979) *The advanced theory of statistics (3 volumes).* Griffin (4th Edition)

5 Parameters

1: WEIGHT — CHARACTER*1 *Input*

On entry: indicates if weights are to be used.

If WEIGHT = 'U', no weights are used.

If WEIGHT = 'W' or 'V', weights are used and must be supplied in WT.

Constraint: WEIGHT = 'U', 'W' or 'V'.

In the case of WEIGHT = 'W', the weights are treated as frequencies and the effective number of observations is the sum of the weights. If WEIGHT = 'V', the weights are treated as being inversely proportional to the variance of the observations and the effective number of observations is the number of observations with non-zero weights.

2: N — INTEGER *Input*

On entry: the number of observations, n.

Constraint: N $\geq$ NX + NG.

3: M — INTEGER *Input*

On entry: the total number of variables, m.

Constraint: M $\geq$ NX.

4: X(LDX,M) — ***real*** array *Input*

On entry: X(i,j) must contain the ith observation for the jth variable, for $i=1,2,\ldots,n$; $j=1,2,\ldots,m$.

5: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03ACF is called.

Constraint: LDX $\geq$ N.

6: ISX(M) — INTEGER array *Input*

On entry: ISX(j) indicates whether or not the jth variable is to be included in the analysis.

If ISX(j) > 0, then the variables contained in the jth column of X is included in the canonical variate analysis, for $j = 1, 2, \ldots, m$.

Constraint: ISX(j) > 0 for NX values of j.

7: NX — INTEGER *Input*

On entry: the number of variables in the analysis, n_x.

Constraint: NX ≥ 1.

8: ING(N) — INTEGER array *Input*

On entry: ING(i) indicates which group the ith observation is in, for $i = 1, 2, \ldots, n$. The effective number of groups is the number of groups with non-zero membership.

Constraint: $1 \leq$ ING(i) $\leq$ NG, for $i = 1, 2, \ldots, n$.

9: NG — INTEGER *Input*

On entry: the number of groups, n_g.

Constraint: NG ≥ 2.

10: WT(*) — ***real*** array *Input*

Note: the dimension of the array WT must be at least N if WEIGHT = 'W' or 'V' and 1 otherwise.

On entry: if WEIGHT = 'W' or 'V', then the first n elements of WT must contain the weights to be used in the analysis.

If WT(i) = 0.0, then the ith observation is not included in the analysis.

If WEIGHT = 'U', then WT is not referenced.

Constraints:

WT(i) ≥ 0.0, for $i = 1, 2, \ldots, n$,

$$\sum_{1}^{n} \mathrm{WT}(i) \geq \mathrm{NX}+ \text{ effective number of groups.}$$

11: NIG(NG) — INTEGER array *Output*

On exit: NIG(j) gives the number of observations in group j, for $j = 1, 2, \ldots, n_g$.

12: CVM(LDCVM,NX) — ***real*** array *Output*

On exit: CVM(i,j) contains the mean of the jth canonical variate for the ith group, for $i = 1, 2, \ldots, n_g$; $j = 1, 2, \ldots, l$; the remaining columns, if any, are used as workspace.

13: LDCVM — INTEGER *Input*

On entry: the dimension of the array CVM as declared in the (sub)program from which G03ACF is called.

Constraint: LDCVM $\geq$ NG.

14: E(LDE,6) — ***real*** array *Output*

On exit: the statistics of the canonical variate analysis.

E(i,1), the canonical correlations, δ_i, for $i = 1,2,\ldots,l$.

E(i,2), the eigenvalues of the within group sum of squares matrix, λ_i^2, for $i = 1,2,\ldots,l$.

E(i,3), the proportion of variation explained by the ith canonical variate, for $i = 1,2,\ldots,l$.

E(i,4), the χ^2 statistic for the ith canonical variate, for $i = 1,2,\ldots,l$.

E(i,5), the degrees of freedom for χ^2 statistic for the ith canonical variate, for $i = 1,2,\ldots,l$.

E(i,6), the significance level for the χ^2 statistic for the ith canonical variate, for $i = 1,2,\ldots,l$.

15: LDE — INTEGER *Input*

On entry: the first dimension of the array E as declared in the (sub)program from which G03ACF is called.

Constraint: LDE $\geq$ min(NX, NG $-$ 1).

16: NCV — INTEGER *Output*

On exit: the number of canonical variates, l. This will be the minimum of $n_g - 1$ and the rank of X.

17: CVX(LDCVX,NG$-$1) — ***real*** array *Output*

On exit: the canonical variate loadings. CVX(i,j) contains the loading coefficient for the ith variable on the jth canonical variate, for $i = 1,2,\ldots,n_x$; $j = 1,2,\ldots,l$; the remaining columns, if any, are used as workspace.

18: LDCVX — INTEGER *Input*

On entry: the first dimension of the array CVX as declared in the (sub)program from which G03ACF is called.

Constraint: LDCVX $\geq$ NX.

19: TOL — ***real*** *Input*

On entry: the value of TOL is used to decide if the variables are of full rank and, if not, what is the rank of the variables. The smaller the value of TOL the stricter the criterion for selecting the singular value decomposition. If a non-negative value of TOL less than ***machine precision*** is entered, then the square root of ***machine precision*** is used instead.

Constraint: TOL $\geq$ 0.0.

20: IRANKX — INTEGER *Output*

On exit: the rank of the dependent variables.

If the variables are of full rank then IRANKX = NX.

If the variables are not of full rank then IRANKX is an estimate of the rank of the dependent variables. IRANK is calculated as the number of singular values greater than TOL$\times$ (largest singular value).

21: WK(IWK) — ***real*** array *Workspace*

22: IWK — INTEGER *Input*

On entry: the dimension of the array WK as declared in the (sub)program from which G03ACF is called.

Constraints:

if NX $\geq$ NG $-$ 1, then IWK $\geq$ N $\times$ NX + max(5 $\times$ (NX $-$ 1) + (NX + 1) $\times$ NX, N),
if NX $<$ NG $-$ 1, then IWK $\geq$ N $\times$ NX + max(5 $\times$ (NX $-$ 1) + (NG $-$ 1) $\times$ NX, N).

23: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, NX < 1,
or NG < 2,
or M < NX,
or N < NX + NG,
or LDX < N,
or LDCVX < NX,
or LDCVM < NG,
or LDE < min(NX,NG−1),
or NX $\geq$ NG−1 and IWK < N × NX + max(5×(NX−1)+(NX+1)×NX,N),
or NX < NG−1 and IWK < N × NX + max(5×(NX−1)+(NG−1)× NX,N),
or WEIGHT $\neq$ 'U', 'W' or 'V',
or TOL < 0.0.

IFAIL = 2

On entry, WEIGHT = 'W' or 'V' and a value of WT < 0.0.

IFAIL = 3

On entry, a value of ING < 1,
or a value of ING > NG.

IFAIL = 4

On entry, the number of variables to be included in the analysis as indicated by ISX is not equal to NX.

IFAIL = 5

A singular value decomposition has failed to converge. This is an unlikely error exit.

IFAIL = 6

A canonical correlation is equal to 1. This will happen if the variables provide an exact indication as to which group every observation is allocated.

IFAIL = 7

On entry, less than two groups have non-zero membership, i.e., the effective number of groups is less than 2,
or the effective number of groups plus the number of variables, NX, is greater than the effective number of observations.

IFAIL = 8

The rank of the variables is 0. This will happen if all the variables are constants.

7 Accuracy

As the computation involves the use of orthogonal matrices and a singular value decomposition rather than the traditional computing of a sum of squares matrix and the use of an eigenvalue decomposition, G03ACF should be less affected by ill conditioned problems.

8 Further Comments

None.

9 Example

A sample of nine observations, each consisting of three variables plus group indicator, is read in. There are three groups. An unweighted canonical variate analysis is performed and the results printed.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03ACF Example Program Text
*     Mark 18 Revised.  NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NMAX, MMAX, IWKMAX
      PARAMETER        (NMAX=9,MMAX=3,IWKMAX=50)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             TOL
      INTEGER          I, IFAIL, IRX, J, M, N, NCV, NG, NX
      CHARACTER        WEIGHT
*     .. Local Arrays ..
      real             CVM(MMAX,MMAX), CVX(MMAX,MMAX), E(MMAX,6),
     +                 WK(IWKMAX), WT(NMAX), X(NMAX,MMAX)
      INTEGER          ING(NMAX), ISX(2*MMAX), NIG(MMAX)
*     .. External Subroutines ..
      EXTERNAL         G03ACF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03ACF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M, NX, NG, WEIGHT
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w' .OR. WEIGHT.EQ.'V' .OR.
     +       WEIGHT.EQ.'v') THEN
            DO 20 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), WT(I), ING(I)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), ING(I)
   40       CONTINUE
         END IF
         READ (5,*) (ISX(J),J=1,M)
         TOL = 0.000001e0
         IFAIL = 0
*
```

```
         CALL G03ACF(WEIGHT,N,M,X,NMAX,ISX,NX,ING,NG,WT,NIG,CVM,MMAX,E,
     +               MMAX,NCV,CVX,MMAX,TOL,IRX,WK,IWKMAX,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Rank of X = ', IRX
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +   'Canonical    Eigenvalues Percentage     CHISQ      DF      SIG'
         WRITE (NOUT,*) 'Correlations              Variation'
         DO 60 I = 1, NCV
            WRITE (NOUT,99998) (E(I,J),J=1,6)
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Canonical Coefficients for X'
         DO 80 I = 1, NX
            WRITE (NOUT,99997) (CVX(I,J),J=1,NCV)
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Canonical variate means'
         DO 100 I = 1, NG
            WRITE (NOUT,99997) (CVM(I,J),J=1,NCV)
  100    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,A,I2)
99998 FORMAT (1X,2F12.4,F11.4,F10.4,F8.1,F8.4)
99997 FORMAT (1X,5F9.4)
      END
```

9.2 Program Data

```
G03ACF Example Program Data
 9 3 3 3 'U'
 13.3 10.6 21.2 1
 13.6 10.2 21.0 2
 14.2 10.7 21.1 3
 13.4  9.4 21.0 1
 13.2  9.6 20.1 2
 13.9 10.4 19.8 3
 12.9 10.0 20.5 1
 12.2  9.9 20.7 2
 13.9 11.0 19.1 3
  1    1    1
```

9.3 Program Results

```
 G03ACF Example Program Results

 Rank of X =  3

 Canonical    Eigenvalues Percentage     CHISQ      DF     SIG
 Correlations              Variation
       0.8826      3.5238     0.9795    7.9032     6.0  0.2453
       0.2623      0.0739     0.0205    0.3564     2.0  0.8368
```

```
Canonical Coefficients for X
  -1.7070   0.7277
  -1.3481   0.3138
   0.9327   1.2199

Canonical variate means
   0.9841   0.2797
   1.1805  -0.2632
  -2.1646  -0.0164
```

G03ADF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G03ADF performs canonical correlation analysis upon input data matrices.

2 Specification

```
      SUBROUTINE G03ADF(WEIGHT, N, M, Z, LDZ, ISZ, NX, NY, WT, E, LDE,
     1                  NCV, CVX, LDCVX, MCV, CVY, LDCVY, TOL, WK, IWK,
     2                  IFAIL)
      INTEGER           N, M, LDZ, ISZ(M), NX, NY, LDE, NCV, LDCVX, MCV,
     1                  LDCVY, IWK, IFAIL
      real              Z(LDZ,M), WT(*), E(LDE,6), CVX(LDCVX,MCV),
     1                  CVY(LDCVY,MCV), TOL, WK(IWK)
      CHARACTER*1       WEIGHT
```

3 Description

Let there be two sets of variables, x and y. For a sample of n observations on n_x variables in a data matrix X and n_y variables in a data matrix Y, canonical correlation analysis seeks to find a small number of linear combinations of each set of variables in order to explain or summarise the relationships between them. The variables thus formed are known as canonical variates.

Let the variance-covariance of the two data sets be

$$\begin{pmatrix} S_{xx} & S_{xy} \\ S_{yx} & S_{yy} \end{pmatrix}$$

and let

$$\Sigma = S_{yy}^{-1} S_{yx} S_{xx}^{-1} S_{xy}$$

then the canonical correlations can be calculated from the eigenvalues of the matrix Σ. However, G03ADF calculates the canonical correlations by means of a singular value decomposition (SVD) of a matrix V. If the rank of the data matrix X is k_x and the rank of the data matrix Y is k_y and both X and Y have had variable (column) means subtracted then the k_x by k_y matrix V is given by:

$$V = Q_x^T Q_y,$$

where Q_x is the first k_x rows of the orthogonal matrix Q either from the QR decomposition of X if X is of full column rank, i.e., $k_x = n_x$:

$$X = Q_x R_x$$

or from the SVD of X if $k_x < n_x$:

$$X = Q_x D_x P_x^T$$

Similarly Q_y is the first k_y rows of the orthogonal matrix Q either from the QR decomposition of Y if Y is of full column rank, i.e., $k_y = n_y$:

$$Y = Q_y R_y$$

or from the SVD of Y if $k_y < n_y$:

$$Y = Q_y D_y P_y^T.$$

Let the SVD of V be:

$$V = U_x \Delta U_y^T$$

then the non-zero elements of the diagonal matrix Δ, δ_i, for $i = 1, 2, \ldots, l$, are the l canonical correlations associated with the l canonical variates, where $l = \min(k_x, k_y)$.

The eigenvalues, λ_i^2, of the matrix Σ are given by:

$$\lambda_i^2 = \frac{\delta_i^2}{1+\delta_i^2}.$$

The value of $\pi_i = \lambda_i^2 / \sum \lambda_i^2$ gives the proportion of variation explained by the ith canonical variate. The values of the π_i's give an indication as to how many canonical variates are needed to adequately describe the data, i.e., the dimensionality of the problem.

To test for a significant dimensionality greater than i the χ^2 statistic:

$$\left(n - \frac{1}{2}(k_x + k_y + 3)\right) \sum_{j=i+1}^{\ell} \log(1+\lambda_j^2)$$

can be used. This is asymptotically distributed as a χ^2 distribution with $(k_x - i)(k_y - i)$ degrees of freedom. If the test for $i = k_{\text{min}}$ is not significant, then the remaining tests for $i > k_{\text{min}}$ should be ignored.

The loadings for the canonical variates are calculated from the matrices U_x and U_y respectively. These matrices are scaled so that the canonical variates have unit variance.

4 References

[1] Chatfield C and Collins A J (1980) *Introduction to Multivariate Analysis* Chapman and Hall

[2] Kendall M G and Stuart A (1976) *The Advanced Theory of Statistics (Volume 3)* Griffin (3rd Edition)

[3] Morrison D F (1967) *Multivariate Statistical Methods* McGraw–Hill

5 Parameters

1: WEIGHT — CHARACTER*1 *Input*

On entry: indicates if weights are to be used.

If WEIGHT = 'U' (Unweighted), no weights are used.

If WEIGHT = 'W' (Weighted), weights are used and must be supplied in WT.

Constraint: WEIGHT = 'U' or 'W'.

2: N — INTEGER *Input*

On entry: the number of observations, n.

Constraint: N > NX + NY.

3: M — INTEGER *Input*

On entry: the total number of variables, m.

Constraint: M $\geq$ NX + NY.

4: Z(LDZ,M) — ***real*** array *Input*

On entry: Z(i,j) must contain the ith observation for the jth variable, for $i = 1,2,\ldots,n$; $j = 1,2,\ldots,m$.

Both x and y variables are to be included in Z, the indicator array, ISZ, being used to assign the variables in Z to the x or y sets as appropriate.

5: LDZ — INTEGER *Input*

On entry: the first dimension of the array Z as declared in the (sub)program from which G03ADF is called.

Constraint: LDZ $\geq$ N.

6: ISZ(M) — INTEGER array *Input*

On entry: ISZ(j) indicates whether or not the jth variable is included in the analysis and to which set of variables it belongs.

If ISZ(j) > 0, then the variable contained in the jth column of Z is included as an x variable in the analysis.

If ISZ(j) < 0, then the variable contained in the jth column of Z is included as a y variable in the analysis.

If ISZ(j) $= 0$, then the variable contained in the jth column of Z is not included in the analysis.

Constraint: only NX elements of ISZ can be > 0 and only NY elements of ISZ can be < 0.

7: NX — INTEGER *Input*

On entry: the number of x variables in the analysis, n_x.

Constraint: NX ≥ 1.

8: NY — INTEGER *Input*

On entry: the number of y variables in the analysis, n_y.

Constraint: NY ≥ 1.

9: WT($*$) — ***real*** array *Input*

On entry: if WEIGHT = 'W', then the first n elements of WT must contain the weights to be used in the analysis.

If WT(i) = 0.0, then the ith observation is not included in the analysis. The effective number of observations is the sum of weights.

If WEIGHT = 'U', then WT is not referenced and the effective number of observations is n.

Constraint: WT(i) ≥ 0.0, for $i = 1, 2, \ldots, n$ and the sum of weights $\geq$ NX + NY + 1.

10: E(LDE,6) — ***real*** array *Output*

On exit: the statistics of the canonical variate analysis.

E(i,1), the canonical correlations, δ_i, for $i = 1, 2, \ldots, l$.

E(i,2), the eigenvalues of Σ, λ_i^2, for $i = 1, 2, \ldots, l$.

E(i,3), the proportion of variation explained by the ith canonical variate, for $i = 1, 2, \ldots, l$.

E(i,4), the χ^2 statistic for the ith canonical variate, for $i = 1, 2, \ldots, l$.

E(i,5), the degrees of freedom for χ^2 statistic for the ith canonical variate, for $i = 1, 2, \ldots, l$.

E(i,6), the significance level for the χ^2 statistic for the ith canonical variate, for $i = 1, 2, \ldots, l$.

11: LDE — INTEGER *Input*

On entry: the first dimension of the array E as declared in the (sub)program from which G03ADF is called.

Constraint: LDE $\geq$ min(NX, NY).

12: NCV — INTEGER *Output*

On exit: the number of canonical correlations, l. This will be the minimum of the rank of X and the rank of Y.

13: CVX(LDCVX,MCV) — ***real*** array *Output*

On exit: the canonical variate loadings for the x variables. CVX(i,j) contains the loading coefficient for the ith x variable on the jth canonical variate.

14: LDCVX — INTEGER *Input*

On entry: the first dimension of the array CVX as declared in the (sub)program from which G03ADF is called.

Constraint: LDCVX $\geq$ NX.

15: MCV — INTEGER *Input*

On entry: an upper limit to the number of canonical variates.

Constraint: MCV $\geq \min(\text{NX}, \text{NY})$.

16: CVY(LDCVY,MCV) — ***real*** array *Output*

On exit: the canonical variate loadings for the y variables. CVY(i,j) contains the loading coefficient for the ith y variable on the jth canonical variate.

17: LDCVY — INTEGER *Input*

On entry: the first dimension of the array CVY as declared in the (sub)program from which G03ADF is called.

Constraint: LDCVY $\geq$ NY.

18: TOL — ***real*** *Input*

On entry: the value of TOL is used to decide if the variables are of full rank and, if not, what is the rank of the variables. The smaller the value of TOL the stricter the criterion for selecting the singular value decomposition. If a non-negative value of TOL less than ***machine precision*** is entered, then the square root of ***machine precision*** is used instead.

Constraint: TOL $\geq$ 0.0.

19: WK(IWK) — ***real*** array *Workspace*

20: IWK — INTEGER *Input*

On entry: the dimension of the array WK as declared in the (sub)program from which G03ADF is called.

Constraints:

if NX $\geq$ NY, then IWK $\geq \text{N} \times \text{NX} + \text{NX} + \text{NY} + \max((5 \times (\text{NX} - 1) + \text{NX} \times \text{NX}), \text{N} \times \text{NY})$,
if NX $<$ NY, then IWK $\geq \text{N} \times \text{NY} + \text{NX} + \text{NY} + \max((5 \times (\text{NY} - 1) + \text{NY} \times \text{NY}), \text{N} \times \text{NX})$.

21: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, $\mathrm{NX} < 1$,
or $\mathrm{NY} < 1$,
or $\mathrm{M} < \mathrm{NX} + \mathrm{NY}$,
or $\mathrm{N} \leq \mathrm{NX} + \mathrm{NY}$,
or $\mathrm{MCV} < \min(\mathrm{NX}, \mathrm{NY})$,
or $\mathrm{LDZ} < \mathrm{N}$,
or $\mathrm{LDCVX} < \mathrm{NX}$,
or $\mathrm{LDCVY} < \mathrm{NY}$,
or $\mathrm{LDE} < \min(\mathrm{NX}, \mathrm{NY})$,
or $\mathrm{NX} \geq \mathrm{NY}$ and $\mathrm{IWK} < \mathrm{N} \times \mathrm{NX} + \mathrm{NX} + \mathrm{NY} + \max((5 \times (\mathrm{NX} - 1) + \mathrm{NX} \times \mathrm{NX}), \mathrm{N} \times \mathrm{NY})$,
or $\mathrm{NX} < \mathrm{NY}$ and $\mathrm{IWK} < \mathrm{N} \times \mathrm{NY} + \mathrm{NX} + \mathrm{NY} + \max((5 \times (\mathrm{NY} - 1) + \mathrm{NY} \times \mathrm{NY}), \mathrm{N} \times \mathrm{NX})$,
or WEIGHT ≠ 'U' or 'W',
or TOL < 0.0.

IFAIL = 2

On entry, a WEIGHT = 'W' and value of WT < 0.0.

IFAIL = 3

On entry, the number of x variables to be included in the analysis as indicated by ISZ is not equal to NX.
or the number of y variables to be included in the analysis as indicated by ISZ is not equal to NY.

IFAIL = 4

On entry, the effective number of observations is less than NX + NY + 1.

IFAIL = 5

A singular value decomposition has failed to converge. See F02WEF or F02WUF. This is an unlikely error exit.

IFAIL = 6

A canonical correlation is equal to 1. This will happen if the x and y variables are perfectly correlated.

IFAIL = 7

On entry, the rank of the X matrix or the rank of the Y matrix is 0. This will happen if all the x or y variables are constants.

7 Accuracy

As the computation involves the use of orthogonal matrices and a singular value decomposition rather than the traditional computing of a sum of squares matrix and the use of an eigenvalue decomposition, G03ADF should be less affected by ill conditioned problems.

8 Further Comments

None.

9 Example

A sample of nine observations with two variables in each set is read in. The second and third variables are x variables while the first and last are y variables. Canonical variate analysis is performed and the results printed.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03ADF Example Program Text
*     Mark 14 Release.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NMAX, IMAX, IWKMAX
      PARAMETER        (NMAX=9,IMAX=2,IWKMAX=40)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             TOL
      INTEGER          I, IFAIL, IX, IY, J, M, N, NCV, NX, NY
      CHARACTER        WEIGHT
*     .. Local Arrays ..
      real             CVX(IMAX,IMAX), CVY(IMAX,IMAX), E(IMAX,6),
     +                 WK(IWKMAX), WT(NMAX), Z(NMAX,2*IMAX)
      INTEGER          ISZ(2*IMAX)
*     .. External Subroutines ..
      EXTERNAL         G03ADF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03ADF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M, IX, IY, WEIGHT
      IF (N.LE.NMAX .AND. IX.LE.IMAX .AND. IY.LE.IMAX) THEN
         IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
            DO 20 I = 1, N
               READ (NIN,*) (Z(I,J),J=1,M), WT(I)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) (Z(I,J),J=1,M)
   40       CONTINUE
         END IF
         READ (5,*) (ISZ(J),J=1,M)
         TOL = 0.000001e0
         NX = IX
         NY = IY
         IFAIL = 0
*
         CALL G03ADF(WEIGHT,N,M,Z,NMAX,ISZ,NX,NY,WT,E,IMAX,NCV,CVX,IMAX,
     +               IMAX,CVY,IMAX,TOL,WK,IWKMAX,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Rank of X = ', NX, ' Rank of Y = ', NY
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +   'Canonical    Eigenvalues Percentage    Chisq      DF      Sig'
         WRITE (NOUT,*) 'correlations                 variation'
         DO 60 I = 1, NCV
            WRITE (NOUT,99998) (E(I,J),J=1,6)
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Canonical coefficients for X'
         DO 80 I = 1, IX
            WRITE (NOUT,99997) (CVX(I,J),J=1,NCV)
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Canonical coefficients for Y'
         DO 100 I = 1, IY
            WRITE (NOUT,99997) (CVY(I,J),J=1,NCV)
  100    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,A,I2,A,I2)
99998 FORMAT (1X,2F12.4,F11.4,F10.4,F8.1,F8.4)
99997 FORMAT (1X,5F9.4)
      END
```

9.2 Program Data

```
G03ADF Example Program Data
 9 4 2 2 'U'
 80.0 58.4 14.0 21.0
 75.0 59.2 15.0 27.0
 78.0 60.3 15.0 27.0
 75.0 57.4 13.0 22.0
 79.0 59.5 14.0 26.0
 78.0 58.1 14.5 26.0
 75.0 58.0 12.5 23.0
 64.0 55.5 11.0 22.0
 80.0 59.2 12.5 22.0
 -1    1    1    -1
```

9.3 Program Results

```
 G03ADF Example Program Results

 Rank of X =  2 Rank of Y =  2

 Canonical    Eigenvalues Percentage     Chisq      DF      Sig
 correlations             variation
       0.9570     10.8916     0.9863   14.3914     4.0  0.0061
       0.3624      0.1512     0.0137    0.7744     1.0  0.3789

 Canonical coefficients for X
   -0.4261   1.0337
   -0.3444  -1.1136
```

```
Canonical coefficients for Y
  -0.1415   0.1504
  -0.2384  -0.3424
```

G03BAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03BAF computes orthogonal rotations for a matrix of loadings using a generalized orthomax criterion.

2. Specification

```
      SUBROUTINE G03BAF (STAND, G, NVAR, K, FL, LDF, FLR, R, LDR, ACC,
     1                   MAXIT, ITER, WK, IFAIL)
      INTEGER            NVAR, K, LDF, LDR, MAXIT, ITER, IFAIL
      real               G, FL(LDF,K), FLR(LDF,K), R(LDR,K), ACC,
     1                   WK(2*NVAR+K*K+5*(K-1))
      CHARACTER*1        STAND
```

3. Description

Let Λ be the p by k matrix of loadings from a variable-directed multivariate method, e.g. canonical variate analysis or factor analysis. This matrix represents the relationship between the original p variables and the k orthogonal linear combinations of these variables, the canonical variates or factors. The latter are only unique up to a rotation in the k-dimensional space they define. A rotation can then be found that simplifies the structure of the matrix of loadings, and hence the relationship between the original and the derived variables. That is the elements, λ_{ij}^*, of the rotated matrix, Λ^*, are either relatively large or small. The rotations may be found by minimizing the criterion:

$$V = \sum_{j=1}^{k}\sum_{i=1}^{p}(\lambda_{ij}^*)^4 - \frac{\gamma}{p}\sum_{j=1}^{k}\left[\sum_{i=1}^{p}(\lambda_{ij}^*)^2\right]^2$$

where the constant γ gives a family of rotations with $\gamma = 1$ giving varimax rotations and $\gamma = 0$ giving quartimax rotations.

It is generally advised that factor loadings should be standardised, so that the sum of squared elements for each row is one, before computing the rotations.

The matrix of rotations, R, such that $\Lambda^* = \Lambda R$, is computed using first an algorithm based on that described by Cooley and Lohnes [1], which involves the pairwise rotation of the factors. Then a final refinement is made using a method similar to that described by Lawley and Maxwell [2], but instead of the eigenvalue decomposition the algorithm has been adapted to incorporate a singular value decomposition.

4. References

[1] COOLEY, W.C. and LOHNES, P.R.
Multivariate Data Analysis.
Wiley, 1971.

[2] LAWLEY, D.N. and MAXWELL, A.E.
Factor Analysis as a Statistical Method.
Butterworths, (2nd Edition) 1971.

5. Parameters

1: STAND – CHARACTER*1. *Input*

On entry: indicates if the matrix of loadings is to be row standardised before rotation.

If STAND = 'S' the loadings are row standardised.

If STAND = 'U' the loadings are left unstandardised.

Constraint: STAND = 'S' or 'U'.

2: G – ***real***. *Input*

On entry: the criterion constant, γ, with $\gamma = 1.0$ giving varimax rotations and $\gamma = 0.0$ giving quartimax rotations.

Constraint: G $\geq$ 0.0.

3: NVAR – INTEGER. *Input*

On entry: the number of original variables, p.

Constraint: NVAR $\geq$ K.

4: K – INTEGER. *Input*

On entry: the number of derived variates or factors, k.

Constraint: K $\geq$ 2.

5: FL(LDF,K) – ***real*** array. *Input/Output*

On entry: the matrix of loadings, Λ. FL(i,j) must contain the loading for the ith variable on the jth factor, for $i = 1,2,...,p$; $j = 1,2,...,k$.

On exit: if STAND = 'S' the elements of FL are standardised so that the sum of squared elements for each row is 1.0 and then after the computation of the rotations are rescaled; this may lead to slight differences between the input and output values of FL. If STAND = 'U' FL will be unchanged on exit.

6: LDF – INTEGER. *Input*

On entry: the first dimension of the arrays FL and FLR as declared in the (sub)program from which G03BAF is called.

Constraint: LDF $\geq$ NVAR.

7: FLR(LDF,K) – ***real*** array. *Output*

On exit: the rotated matrix of loadings, Λ^*. FLR(i,j) will contain the rotated loading for the ith variable on the jth factor, for $i = 1,2,...,p$; $j = 1,2,...,k$.

8: R(LDR,K) – ***real*** array. *Output*

On exit: the matrix of rotations, R.

9: LDR – INTEGER. *Input*

On entry: the first dimension of the array R as declared in the (sub)program from which G03BAF is called.

Constraint: LDR $\geq$ K.

10: ACC – ***real***. *Input*

On entry: indicates the accuracy required. The iterative procedure of Cooley and Lohnes [1] will be stopped and the final refinement computed when the change in V is less than ACC×max(1.0,V). If ACC is greater than or equal to 0.0 but less than ***machine precision*** or if ACC is greater than 1.0, then ***machine precision*** will be used instead.

Suggested value: 0.00001.

Constraint: ACC $\geq$ 0.0.

11: MAXIT – INTEGER. *Input*

On entry: the maximum number of iterations.

Suggested value: 30.

Constraint: MAXIT $\geq$ 1.

12: ITER – INTEGER. *Output*

On exit: the number of iterations performed.

13: WK(2*NVAR+K*K+5*(K–1)) – ***real*** array. *Workspace*

14: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, K < 2,
or NVAR < K,
or G < 0.0,
or LDF < NVAR,
or LDR < K,
or ACC < 0.0,
or MAXIT $\leq$ 0,
or STAND $\neq$ 'S' or 'U'.

IFAIL = 2

The singular value decomposition has failed to converge. This is an unlikely error exit.

IFAIL = 3

The algorithm to find R has failed to reach the required accuracy in the given number of iterations. The user should try increasing ACC or increasing MAXIT. The returned solution should be a reasonable approximation.

7. Accuracy

The accuracy is determined by the value of ACC.

8. Further Comments

If the results of a principal component analysis as carried out by G03AAF are to be rotated, the loadings as returned in the array P by G03AAF can be supplied via the parameter FL to G03BAF. The resulting rotation matrix can then be used to rotate the principal component scores as returned in the array V by G03AAF. The routine F06YAF may be used for this matrix multiplication.

9. Example

The example is taken from Lawley and Maxwell [2] (page 75). The results from a factor anaysis of ten variables using three factors are input and rotated using varimax rotations without standardising rows.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03BAF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX
      PARAMETER        (NMAX=10,MMAX=3)
*     .. Local Scalars ..
      real             ACC, G
      INTEGER          I, IFAIL, ITER, J, K, MAXIT, NVAR
      CHARACTER        STAND
*     .. Local Arrays ..
      real             FL(NMAX,MMAX), FLR(NMAX,MMAX), R(MMAX,MMAX),
     +                 WK(2*NMAX+MMAX*MMAX+5*(MMAX-1))
*     .. External Subroutines ..
      EXTERNAL         G03BAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03BAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) NVAR, K, G, STAND, ACC, MAXIT
      IF (NVAR.LE.NMAX .AND. K.LE.MMAX) THEN
         DO 20 I = 1, NVAR
            READ (NIN,*) (FL(I,J),J=1,K)
   20    CONTINUE
         IFAIL = 0
*
         CALL G03BAF(STAND,G,NVAR,K,FL,NMAX,FLR,R,MMAX,ACC,MAXIT,ITER,
     +               WK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) '    Rotated factor loadings'
         WRITE (NOUT,*)
         DO 40 I = 1, NVAR
            WRITE (NOUT,99999) (FLR(I,J),J=1,K)
   40    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) '    Rotation matrix'
         WRITE (NOUT,*)
         DO 60 I = 1, K
            WRITE (NOUT,99999) (R(I,J),J=1,K)
   60    CONTINUE
      END IF
      STOP
*
99999 FORMAT (4(2X,6F8.3))
      END
```

9.2. Program Data

```
G03BAF Example Program Data
  10 3 1.0 'U' 0.00001 20
 0.788 -0.152 -0.352
 0.874  0.381  0.041
 0.814 -0.043 -0.213
 0.798 -0.170 -0.204
 0.641  0.070 -0.042
 0.755 -0.298  0.067
 0.782 -0.221  0.028
 0.767 -0.091  0.358
 0.733 -0.384  0.229
 0.771 -0.101  0.071
```

9.3. Program Results

```
 G03BAF Example Program Results

     Rotated factor loadings

     0.329  -0.289  -0.759
     0.849  -0.273  -0.340
     0.450  -0.327  -0.633
     0.345  -0.397  -0.657
     0.453  -0.276  -0.370
     0.263  -0.615  -0.464
     0.332  -0.561  -0.485
     0.472  -0.684  -0.183
     0.209  -0.754  -0.354
     0.423  -0.514  -0.409

     Rotation matrix

     0.633  -0.534  -0.560
     0.758   0.573   0.311
     0.155  -0.622   0.768
```

G03BCF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G03BCF computes Procrustes rotations in which an orthogonal rotation is found so that a transformed matrix best matches a target matrix.

2 Specification

```
      SUBROUTINE G03BCF(STAND, PSCALE, N, M, X, LDX, Y, LDY, YHAT, R,
     1                  LDR, ALPHA, RSS, RES, WK, IFAIL)
      INTEGER           N, M, LDX, LDY, LDR, IFAIL
      real              X(LDX,M), Y(LDY,M), YHAT(LDY,M), R(LDR,M),
     1                  ALPHA, RSS, RES(N), WK(M*M+7*M)
      CHARACTER*1       STAND, PSCALE
```

3 Description

Let X and Y be n by m matrices. They can be considered as representing sets of n points in an m-dimensional space. The X matrix may be a matrix of loadings from say factor analysis or canonical variate analysis and the Y matrix may be a postulated pattern matrix or the loadings from a different sample. The problem is to relate the two sets of points without disturbing the relationships between the points in each set. This can be achieved by translating, rotating and scaling the sets of points. The Y matrix is considered as the target matrix and the X matrix is rotated to match that matrix.

First the two sets of points are translated so that their centroids are at the origin to give X_c and Y_c, i.e., the matrices will have zero column means. Then the rotation of the translated X_c matrix which minimizes the sum of squared distances between corresponding points in the two sets is found. This is computed from the singular value decomposition of the matrix:

$$X_c^T Y_c = UDV^T,$$

where U and V are orthogonal matrices and D is a diagonal matrix. The matrix of rotations, R, is computed as:

$$R = UV^T.$$

After rotation a scaling or dilation factor, α, may be estimated by least-squares. Thus the final set of points that best match Y_c is given by:

$$\hat{Y}_c = \alpha X_c R.$$

Before rotation both sets of points may be normalized to have unit sums of squares or the X matrix may be normalized to have the same sum of squares as the Y matrix. After rotation the results may be translated to the original Y centroid.

The ith residual, r_i, is given by the distance between the point given in the ith row of Y and the point given in the ith row of $\hat{Y}$. The residual sum of squares is also computed.

4 References

[1] Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

[2] Lawley D N and Maxwell A E (1971) *Factor Analysis as a Statistical Method* Butterworths (2nd Edition)

5 Parameters

1: STAND — CHARACTER*1 *Input*

On entry: indicates if translation/normalization is required.

If STAND = 'N' there is no translation or normalization.

If STAND = 'Z' there is translation to the origin (i.e., to zero).

If STAND = 'C' there is translation to origin and then to the Y centroid after rotation.

If STAND = 'U' there is unit normalization.

If STAND = 'S' there is translation and normalization (i.e., there is standardization).

If STAND = 'M' there is translation and normalization to Y scale, then translation to the Y centroid after rotation (i.e., they are matched).

Constraint: STAND = 'N', 'Z', 'C', 'U', 'S' or 'M'.

2: PSCALE — CHARACTER*1 *Input*

On entry: indicates if least-squares scaling is to be applied after rotation.

If PSCALE = 'S' , then scaling is applied.

If PSCALE = 'U', then no scaling is applied.

Constraint: PSCALE = 'S' or 'U'.

3: N — INTEGER *Input*

On entry: the number of points, n.

Constraint: N $\geq$ M.

4: M — INTEGER *Input*

On entry: the number of dimensions, m.

Constraint: M $\geq$ 1.

5: X(LDX,M) — ***real*** array *Input/Output*

On entry: the matrix to be rotated, X.

On exit: if STAND = 'N', then X will be unchanged.

If STAND = 'Z', 'C', 'S' or 'M', then X will be translated to have zero column means.

If STAND = 'U' or 'S', then X will be scaled to have unit sum of squares.

If STAND = 'M', then X will be scaled to have the same sum of squares as Y.

6: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03BCF is called.

Constraint: LDX $\geq$ N.

7: Y(LDY,M) — ***real*** array *Input/Output*

On entry: the target matrix, Y.

On exit: if STAND = 'N', then Y will be unchanged.

If STAND = 'Z' or 'S', then Y will be translated to have zero column means.

If STAND = 'U' or 'S', then Y will be scaled to have unit sum of squares.

If STAND = 'C' or 'M', then Y will be translated and then after rotation translated back. The output Y should be the same as the input Y except for rounding errors.

8: LDY — INTEGER *Input*

On entry: the first dimension of the arrays Y and YHAT as declared in the (sub)program from which G03BCF is called.

Constraint: LDY $\geq$ N.

9: YHAT(LDY,M) — ***real*** array *Output*

On exit: the fitted matrix, $\hat{Y}$.

10: R(LDR,M) — ***real*** array *Output*

On exit: the matrix of rotations, R, see Section 8.

11: LDR — INTEGER *Input*

On entry: the first dimension of the array R as declared in the (sub)program from which G03BCF is called.

Constraint: LDR $\geq$ M.

12: ALPHA — ***real*** *Output*

On exit: if PSCALE = 'S' the scaling factor, α; otherwise ALPHA is not set.

13: RSS — ***real*** *Output*

On exit: the residual sum of squares.

14: RES(N) — ***real*** array *Output*

On exit: the residuals, r_i, for $i = 1,2,$.

15: WK(M*M+7*M) — ***real*** array *Workspace*

16: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, N < M,
or M < 1,
or LDX < N,
or LDY < N,
or LDR < M,
or STAND ≠ 'N', 'Z', 'C', 'U', 'S' or 'M',
or PSCALE ≠ 'S' or 'U'.

IFAIL = 2

On entry, either X or Y contain only zero-points (possibly after translation) when normalization is to be applied.

IFAIL = 3

The $\hat{Y}$ matrix contains only zero-points when least-squares scaling is applied.

IFAIL = 4

The singular value decomposition has failed to converge. This is an unlikely error exit.

7 Accuracy

The accuracy of the calculation of the rotation matrix largely depends upon the singular value decomposition. See F02WEF for further details.

8 Further Comments

Note that if the matrix $X_c^T Y$ is not of full rank, then the matrix of rotations, R, may not be unique even if there is a unique solution in terms of the rotated matrix, $\hat{Y}_c$. The matrix R may also include reflections as well as pure rotations, see Krzanowski [1].

If the column dimensions of the X and Y matrices are not equal, the smaller of the two should be supplemented by columns of zeros. Adding a column of zeros to both X and Y will have the effect of allowing reflections as well as rotations.

9 Example

Three points representing the vertices of a triangle in two dimensions are input. The points are translated and rotated to match the triangle given by (0,0),(1,0),(0,2) and scaling is applied after rotation. The target matrix and fitted matrix are printed along with additional information.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03BCF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX
      PARAMETER        (NMAX=3,MMAX=2)
*     .. Local Scalars ..
      real             ALPHA, RSS
      INTEGER          I, IFAIL, J, M, N
      CHARACTER        SCALE, STAND
*     .. Local Arrays ..
      real             R(MMAX,MMAX), RES(NMAX), WK(MMAX*MMAX+7*MMAX),
     +                 X(NMAX,MMAX), Y(NMAX,MMAX), YHAT(NMAX,MMAX)
*     .. External Subroutines ..
      EXTERNAL         G03BCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03BCF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M, STAND, SCALE
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         DO 20 I = 1, N
            READ (NIN,*) (X(I,J),J=1,M)
   20    CONTINUE
         DO 40 I = 1, N
            READ (NIN,*) (Y(I,J),J=1,M)
   40    CONTINUE
         IFAIL = 0
*
         CALL G03BCF(STAND,SCALE,N,M,X,NMAX,Y,NMAX,YHAT,R,MMAX,ALPHA,
     +               RSS,RES,WK,IFAIL)
```

```
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) '         Rotation Matrix'
         WRITE (NOUT,*)
         DO 60 I = 1, M
            WRITE (NOUT,99999) (R(I,J),J=1,M)
   60    CONTINUE
         IF (SCALE.EQ.'S' .OR. SCALE.EQ.'s') THEN
            WRITE (NOUT,*)
            WRITE (NOUT,99998) ' Scale factor = ', ALPHA
         END IF
         WRITE (NOUT,*)
         WRITE (NOUT,*) '        Target Matrix'
         WRITE (NOUT,*)
         DO 80 I = 1, N
            WRITE (NOUT,99999) (Y(I,J),J=1,M)
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) '        Fitted Matrix'
         WRITE (NOUT,*)
         DO 100 I = 1, N
            WRITE (NOUT,99999) (YHAT(I,J),J=1,M)
  100    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'RSS = ', RSS
      END IF
      STOP
*
99999 FORMAT (6(2X,F7.3))
99998 FORMAT (1X,A,F10.3)
      END
```

9.2 Program Data

```
G03BCF EXAMPLE PROGRAM DATA
3 2 'c' 's'
0.63 0.58
1.36 0.39
1.01 1.76
0.0 0.0
1.0 0.0
0.0 2.0
```

9.3 Program Results

```
G03BCF Example Program Results

         Rotation Matrix

   0.967    0.254
  -0.254    0.967

 Scale factor =      1.556

        Target Matrix

   0.000    0.000
```

```
      1.000     0.000
      0.000     2.000

           Fitted Matrix

     -0.093     0.024
      1.080     0.026
      0.013     1.950

   RSS =       0.019
```

G03CAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03CAF computes the maximum likelihood estimates of the parameters of a factor analysis model. Either the data matrix or a correlation/covariance matrix may be input. Factor loadings, communalities and residual correlations are returned.

2. Specification

```
 SUBROUTINE G03CAF (MATRIX, WEIGHT, N, M, X, LDX, NVAR, ISX,
1                   NFAC, WT, E, STAT, COM, PSI, RES, FL, LDFL,
2                   IOP, IWK, WK, LWK, IFAIL)
 INTEGER        N, M, LDX, NVAR, ISX(M), NFAC, LDFL, IOP(5),
1               IWK(4*NVAR+2), LWK, IFAIL
 real           X(LDX,M), WT(*), E(NVAR), STAT(4), COM(NVAR),
1               PSI(NVAR), RES(NVAR*(NVAR-1)/2), FL(LDFL,NFAC),
2               WK(LWK)
 CHARACTER*1    MATRIX, WEIGHT
```

3. Description

Let p variables, $x_1,x_2,...,x_p$, with variance-covariance matrix Σ be observed. The aim of factor analysis is to account for the covariances in these p variables in terms of a smaller number, k, of hypothetical variables, or factors, $f_1,f_2,...,f_k$. These are assumed to be independent and to have unit variance. The relationship between the observed variables and the factors is given by the model:

$$x_i = \sum_{j=1}^{k} \lambda_{ij} f_j + e_i \quad i = 1,2,...,p$$

where λ_{ij}, for $i = 1,2,...,p$; $j = 1,2,...,k$, are the factor loadings and e_i, for $i = 1,2,...,p$, are independent random variables with variances ψ_i, for $i = 1,2,...,p$. The ψ_i represent the unique component of the variation of each observed variable. The proportion of variation for each variable accounted for by the factors is known as the communality. For this routine it is assumed that both the k factors and the e_i's follow independent normal distributions.

The model for the variance-covariance matrix, Σ, can be written as:

$$\Sigma = \Lambda\Lambda^T + \Psi, \tag{1}$$

where Λ is the matrix of the factor loadings, λ_{ij}, and Ψ is a diagonal matrix of unique variances, ψ_i, for $i = 1,2,...,p$.

The estimation of the parameters of the model, Λ and Ψ, by maximum likelihood is described by Lawley and Maxwell [2]. The log likelihood is:

$$-\tfrac{1}{2}(n-1)\log(|\Sigma|) - \tfrac{1}{2}(n-1)\mathrm{trace}(S\Sigma^{-1}) + \text{constant},$$

where n is the number of observations, S is the sample variance-covariance matrix or if weights are used S is the weighted sample variance-covariance matrix and n is the effective number of observations, that is the sum of the weights. The constant is independent of the parameters of the model. A two stage maximization is employed. It makes use of the function $F(\Psi)$, which is, up to a constant, $-2/(n-1)$ times the log likelihood maximized over Λ. This is then minimized with respect to Ψ to give the estimates, $\hat{\Psi}$, of Ψ. The function $F(\Psi)$ can be written as:

$$F(\Psi) = \sum_{j=k+1}^{p} (\theta_j - \log\theta_j) - (p-k) ,$$

where values θ_j, for $j = 1,2,...,p$ are the eigenvalues of the matrix:

$$S^* = \Psi^{-1/2} S \Psi^{-1/2}.$$

The estimates $\hat{\Lambda}$, of Λ, are then given by scaling the eigenvectors of S^*, which are denoted by V:

$$\hat{\Lambda} = \Psi^{1/2}V(\Theta-I)^{1/2},$$

where Θ is the diagonal matrix with elements θ_i, and I is the identity matrix.

The minimization of $F(\Psi)$ is performed using E04LBF which uses a modified Newton algorithm. The computation of the Hessian matrix is described by Clarke [1]. However, instead of using the eigenvalue decomposition of the matrix S^* as described above the singular value decomposition of the matrix $R\Psi^{-1/2}$ is used, where R is obtained either from the QR decomposition of the (scaled) mean centred data matrix or from the Cholesky decomposition of the correlation/covariance matrix. The routine E04LBF ensures that the values of ψ_i are greater than a given small positive quantity, δ, so that the communality is always less than one. This avoids the so called Heywood cases.

In addition to the values of Λ, Ψ and the communalities, G03CAF returns the residual correlations, i.e. the off-diagonal elements of $C - (\Lambda\Lambda^T+\Psi)$ where C is the sample correlation matrix. G03CAF also returns the test statistic:

$$\chi^2 = [n-1-(2p+5)/6-2k/3]F(\hat{\Psi})$$

which can be used to test the goodness of fit of the model (1), see Lawley and Maxwell [2] and Morrison [4].

4. References

[1] CLARKE, M.R.B.
A Rapidly Convergent Method for Maximum Likelihood Factor Analysis.
Br. J. Math. Statist. Psych., 1970.

[2] LAWLEY, D.N. and MAXWELL, A.E.
Factor Analysis as a Statistical Method.
Butterworths, (2nd Edition) 1971.

[3] HAMMARLING, S.
The Singular Value Decomposition in Multivariate Statistics.
Signum Newsletter, 20, pp. 2-25, 1985.

[4] MORRISON, D.F.
Multivariate Statistical Methods.
McGraw-Hill, 1967.

5. Parameters

1: MATRIX – CHARACTER*1. *Input*

On entry: selects the type of matrix on which factor analysis is to be performed.

If MATRIX = 'D' (Data input), then the data matrix will be input in X and factor analysis will be computed for the correlation matrix.

If MATRIX = 'S', then the data matrix will be input in X and factor analysis will be computed for the covariance matrix, i.e. the results are scaled as described in Section 8.

If MATRIX = 'C', then the correlation/variance-covariance matrix will be input in X and factor analysis computed for this matrix.

See Section 8 for further comments.

Constraint: MATRIX = 'D', 'S' or 'C'.

2: WEIGHT – CHARACTER*1. *Input*

On entry: if MATRIX = 'D' or 'S', WEIGHT indicates if weights are to be used.

If WEIGHT = 'U', then no weights are used.

If WEIGHT = 'W', then weights are used and must be supplied in WT.

Note: if MATRIX = 'C', WEIGHT is not referenced.

Constraint: if MATRIX = 'D' or 'S', WEIGHT = 'U' or 'W'.

3: N – INTEGER. *Input*

On entry: if MATRIX = 'D' or 'S' the number of observations in the data array X.

If MATRIX = 'C' the (effective) number of observations used in computing the (possibly weighted) correlation/variance-covariance matrix input in X.

Constraint: N > NVAR.

4: M – INTEGER. *Input*

On entry: the number of variables in the data/correlation/variance-covariance matrix.

Constraint: M ≥ NVAR.

5: X(LDX,M) – ***real*** array. *Input*

On entry: the input matrix.

If MATRIX = 'D' or 'S', then X must contain the data matrix, i.e. X(i,j) must contain the ith observation for the jth variable, for $i = 1,2,...,n$; $j = 1,2,...,$M.

If MATRIX = 'C', then X must contain the correlation or variance-covariance matrix. Only the upper triangular part is required.

6: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03CAF is called.

Constraints: if MATRIX = 'D' or 'S', then LDX ≥ N,
if MATRIX = 'C', then LDX ≥ M.

7: NVAR – INTEGER. *Input*

On entry: the number of variables in the factor analysis, p.

Constraint: NVAR ≥ 2.

8: ISX(M) – INTEGER array. *Input*

On entry: ISX(j) indicates whether or not the jth variable is included in the factor analysis. If ISX(j) ≥ 1, then the variable represented by the jth column of X is included in the analysis; otherwise it is excluded, for $j = 1,2,...,$M.

Constraint: ISX(j) > 0 for NVAR values of j.

9: NFAC – INTEGER. *Input*

On entry: the number of factors, k.

Constraint: 1 ≤ NFAC ≤ NVAR.

10: WT(*) – ***real*** array. *Input*

On entry: if WEIGHT = 'W' and MATRIX = 'D' or 'S', WT must contain the weights to be used in the factor analysis. The effective number of observations in the analysis will then be the sum of weights. If WT(i) = 0.0, then the ith observation is not included in the analysis.

If WEIGHT = 'U' or MATRIX = 'C', WT is not referenced and the effective number of observations is n.

Constraint: if WEIGHT = 'W', then WT(i) ≥ 0.0, for $i = 1,2,...,n$, and the sum of weights > NVAR.

11: E(NVAR) – ***real*** array. *Output*

On exit: the eigenvalues θ_i, for $i = 1,2,...,p$.

12: STAT(4) – ***real*** array. *Output*

On exit: the test statistics.

STAT(1) contains the value $F(\hat{\Psi})$.

STAT(2) contains the test statistic, χ^2.

STAT(3) contains the degrees of freedom associated with the test statistic.

STAT(4) contains the significance level.

13: COM(NVAR) – ***real*** array. *Output*

On exit: the communalities.

14: PSI(NVAR) – ***real*** array. *Output*

On exit: the estimates of ψ_i, for $i = 1,2,...,p$.

15: RES(NVAR*(NVAR–1)/2) – ***real*** array. *Output*

On exit: the residual correlations. The residual correlation for the ith and jth variables is stored in RES($(j-1)(j-2)/2+i$), $i < j$.

16: FL(LDFL,NFAC) – ***real*** array. *Output*

On exit: the factor loadings. FL(i,j) contains λ_{ij}, for $i = 1,2,...,p$; $j = 1,2,...,k$.

17: LDFL – INTEGER. *Input*

On entry: the first dimension of the array FL as declared in the (sub)program from which G03CAF is called.

Constraint: LDFL $\geq$ NVAR.

18: IOP(5) – INTEGER array. *Input*

On entry: options for the optimization. There are four options to be set:

iprint – controls iteration monitoring;

if *iprint* ≤ 0, then there is no printing of information else if *iprint* > 0, then information is printed at every *iprint* iterations. The information printed consists of the value of $F(\Psi)$ at that iteration, the number of evaluations of $F(\Psi)$, the current estimates of the communalities and an indication of whether or not they are at the boundary.

maxfun – the maximum number of function evaluations.

acc – the required accuracy for the estimates of ψ_i.

eps – a lower bound for the values of ψ, see Section 3.

Let ε = ***machine precision*** then if IOP(1) = 0, then the following default values are used:

$$\begin{aligned} iprint &= -1 \\ maxfun &= 100p \\ acc &= 10\sqrt{\varepsilon} \\ eps &= \varepsilon \end{aligned}$$

If IOP(1) $\neq 0$, then

$$\begin{aligned} iprint &= \text{IOP}(2) \\ maxfun &= \text{IOP}(3) \\ acc &= 10^{-l} \text{ where } l = \text{IOP}(4) \\ eps &= 10^{-l} \text{ where } l = \text{IOP}(5) \end{aligned}$$

Constraint: if IOP(1) $\neq 0$, then IOP(i), for $i = 3,4,5$ must be such that *maxfun* ≥ 1, $\varepsilon \leq acc < 1.0$ and $\varepsilon \leq eps < 1.0$.

19: IWK(4*NVAR+2) – INTEGER array. *Workspace*

20: WK(LWK) – ***real*** array. *Workspace*
21: LWK – INTEGER. *Input*

On entry: the length of the workspace.

Constraints: if MATRIX = 'D' or 'S', then
LWK ≥ max((5×NVAR×NVAR+33×NVAR−4)/2,
NxNVAR + 7xNVAR + NVARx(NVAR−1)/2).

If MATRIX = 'C', then
LWK ≥ (5×NVAR×NVAR+33×NVAR−4)/2.

22: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL ≠ 0 on exit, users are recommended to set IFAIL to −1 before entry. **It is then essential to test the value of IFAIL on exit.**

6. Error Indicators and Warnings

Errors or warnings specified by the routine:

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, LDFL < NVAR,
or NVAR < 2,
or N ≤ NVAR,
or NFAC < 1,
or NVAR < NFAC,
or M < NVAR,
or MATRIX = 'D' or 'S' and LDX < N,
or MATRIX = 'C' and LDX < M,
or MATRIX ≠ 'D' 'S' or 'C',
or MATRIX = 'D' or 'S' and WEIGHT ≠ 'U' or 'W',
or IOP(1) ≠ 0 and IOP(3) is such that *maxfun* < 1,
or IOP(1) ≠ 0 and IOP(4) is such that *acc* ≥ 1.0,
or IOP(1) ≠ 0 and IOP(4) is such that *acc* < ***machine precision***,
or IOP(1) ≠ 0 and IOP(5) is such that *eps* ≥ 1.0,
or IOP(1) ≠ 0 and IOP(5) is such that *eps* < ***machine precision***,
or MATRIX = 'C' and LWK < (5×NVAR×NVAR+33×NVAR−4)/2,
or MATRIX = 'D' or 'S' and
LWK < max((5×NVAR×NVAR+33×NVAR−4)/2,
N×NVAR + 7×NVAR + NVAR×(NVAR−1)/2).

IFAIL = 2

On entry, WEIGHT = 'W' and a value of WT < 0.0.

IFAIL = 3

On entry, there are not exactly NVAR elements of ISX > 0, or the effective number of observations ≤ NVAR.

IFAIL = 4

On entry, MATRIX = 'D' or 'S' and the data matrix is not of full column rank, or MATRIX = 'C' and the input correlation/variance-covariance matrix is not positive definite.

This exit may also be caused by two of the eigenvalues of S^* being equal; this is rare (see [2]), and may be due to the data/correlation matrix being almost singular.

IFAIL = 5

A singular value decomposition has failed to converge. This is a very unlikely error exit.

IFAIL = 6

The estimation procedure has failed to converge in the given number of iterations. Change IOP to either increase number of iterations *maxfun* or increase the value of *acc*.

IFAIL = 7

The convergence is not certain but a lower point could not be found. See E04LBF for further details. In this case all results are computed.

7. Accuracy

The accuracy achieved is discussed in E04LBF with the value of the parameter XTOL given by *acc* as described in Section 5.

8. Further Comments

The factor loadings may be orthogonally rotated by using G03BAF and factor score coefficients can be computed using G03CCF.

The maximum likelihood estimators are invariant to a change in scale. This means that the results obtained will be the same (up to a scaling factor) if either the correlation matrix or the variance-covariance matrix is used. As the correlation matrix ensures that all values of ψ_i are between 0 and 1 it will lead to a more efficient optimization. In the situation when the data matrix is input the results are always computed for the correlation matrix and then scaled if the results for the covariance matrix are required. When the user inputs the covariance/correlation matrix the input matrix itself is used and so the user is advised to input the correlation matrix rather than the covariance matrix.

9. Example

The example is taken from Lawley and Maxwell [2]. The correlation matrix for nine variables is input and the parameters of a factor analysis model with three factors are estimated and printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03CAF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, LWK
      PARAMETER        (NMAX=9,MMAX=9,LWK=349)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, L, M, N, NFAC, NVAR
      CHARACTER        MATRIX, WEIGHT
*     .. Local Arrays ..
      real             COM(MMAX), E(MMAX), FL(MMAX,MMAX), PSI(MMAX),
     +                 RES(MMAX*(MMAX-1)/2), STAT(4), WK(LWK), WT(NMAX),
     +                 X(NMAX,MMAX)
      INTEGER          IOP(5), ISX(MMAX), IWK(4*MMAX+2)
```

```
*     .. External Subroutines ..
      EXTERNAL         G03CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03CAF Example Program Results'
*     Skip headings in data file
      READ (NIN,*)
      READ (NIN,*) MATRIX, WEIGHT, N, M, NVAR, NFAC
      IF (M.LE.MMAX .AND. (MATRIX.EQ.'C' .OR. MATRIX.EQ.'c' .OR. N.LE.
     +    NMAX)) THEN
         IF (MATRIX.EQ.'C' .OR. MATRIX.EQ.'c') THEN
            DO 20 I = 1, M
               READ (NIN,*) (X(I,J),J=1,M)
   20       CONTINUE
         ELSE
            IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
               DO 40 I = 1, N
                  READ (NIN,*) (X(I,J),J=1,M), WT(I)
   40          CONTINUE
            ELSE
               DO 60 I = 1, N
                  READ (NIN,*) (X(I,J),J=1,M)
   60          CONTINUE
            END IF
         END IF
         READ (NIN,*) (ISX(J),J=1,M)
         READ (NIN,*) (IOP(J),J=1,5)
         IFAIL = -1
*
         CALL G03CAF(MATRIX,WEIGHT,N,M,X,NMAX,NVAR,ISX,NFAC,WT,E,STAT,
     +               COM,PSI,RES,FL,MMAX,IOP,IWK,WK,LWK,IFAIL)
*
         IF (IFAIL.EQ.0 .OR. IFAIL.GT.4) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Eigenvalues'
            WRITE (NOUT,*)
            WRITE (NOUT,99998) (E(J),J=1,M)
            WRITE (NOUT,*)
            WRITE (NOUT,99997) '     Test Statistic = ', STAT(2)
            WRITE (NOUT,99997) '                 df = ', STAT(3)
            WRITE (NOUT,99997) ' Significance level = ', STAT(4)
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Residuals'
            WRITE (NOUT,*)
            L = 1
            DO 80 I = 1, NVAR - 1
               WRITE (NOUT,99999) (RES(J),J=L,L+I-1)
               L = L + I
   80       CONTINUE
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Loadings, Communalities and PSI'
            WRITE (NOUT,*)
            DO 100 I = 1, NVAR
               WRITE (NOUT,99999) (FL(I,J),J=1,NFAC), COM(I), PSI(I)
  100       CONTINUE
         END IF
      END IF
      STOP
*
99999 FORMAT (2X,9F8.3)
99998 FORMAT (2X,6e12.4)
99997 FORMAT (A,F6.3)
      END
```

9.2. Program Data

```
G03CAF Example Program Data
'C' 'U' 211 9  9 3
 1.000 0.523 0.395 0.471 0.346 0.426 0.576 0.434 0.639
 0.523 1.000 0.479 0.506 0.418 0.462 0.547 0.283 0.645
 0.395 0.479 1.000 0.355 0.270 0.254 0.452 0.219 0.504
 0.471 0.506 0.355 1.000 0.691 0.791 0.443 0.285 0.505
 0.346 0.418 0.270 0.691 1.000 0.679 0.383 0.149 0.409
 0.426 0.462 0.254 0.791 0.679 1.000 0.372 0.314 0.472
 0.576 0.547 0.452 0.443 0.383 0.372 1.000 0.385 0.680
 0.434 0.283 0.219 0.285 0.149 0.314 0.385 1.000 0.470
 0.639 0.645 0.504 0.505 0.409 0.472 0.680 0.470 1.000
  1    1    1    1    1    1    1    1    1
1 -1 500 2 5
```

9.3. Program Results

```
 G03CAF Example Program Results

  Eigenvalues

    0.1597E+02  0.4358E+01  0.1847E+01  0.1156E+01  0.1119E+01  0.1027E+01
    0.9257E+00  0.8951E+00  0.8771E+00

     Test Statistic =  7.149
                 df = 12.000
 Significance level =  0.848

  Residuals

     0.000
    -0.013   0.022
     0.011  -0.005   0.023
    -0.010  -0.019  -0.016   0.003
    -0.005   0.011  -0.012  -0.001  -0.001
     0.015  -0.022  -0.011   0.002   0.029  -0.012
    -0.001  -0.011   0.013   0.005  -0.006  -0.001   0.003
    -0.006   0.010  -0.005  -0.011   0.002   0.007   0.003  -0.001

  Loadings, Communalities and PSI

     0.664  -0.321   0.074   0.550   0.450
     0.689  -0.247  -0.193   0.573   0.427
     0.493  -0.302  -0.222   0.383   0.617
     0.837   0.292  -0.035   0.788   0.212
     0.705   0.315  -0.153   0.619   0.381
     0.819   0.377   0.105   0.823   0.177
     0.661  -0.396  -0.078   0.600   0.400
     0.458  -0.296   0.491   0.538   0.462
     0.766  -0.427  -0.012   0.769   0.231
```

If the input value of IOP(2) is changed from –1 to 1 the following results are obtained.

```
 G03CAF Example Program Results

 Iterations performed =              0, function evaluations =             1
 Criterion =      0.863576E-01

                 Variable    Standardized
                             Communalities
                     1           0.5755
                     2           0.5863
                     3           0.4344
                     4           0.7496
                     5           0.6203
                     6           0.7329
                     7           0.6061
                     8           0.4053
                     9           0.7104

 Iterations performed =              1, function evaluations =             3
 Criterion =     0.360320E-01

             Variable      Standardized
                           Communalities
                 1             0.5517
                 2             0.5800
                 3             0.3936
                 4             0.7926
                 5             0.6140
                 6             0.8254
                 7             0.6052
                 8             0.5076
                 9             0.7569

 Iterations performed =              2, function evaluations =                4
 Criterion =     0.350210E-01

             Variable      Standardized
                           Communalities
                 1             0.5496
                 2             0.5731
                 3             0.3838
                 4             0.7875
                 5             0.6200
                 6             0.8238
                 7             0.6006
                 8             0.5349
                 9             0.7697

 Iterations performed =              3, function evaluations =                5
 Criterion =     0.350173E-01

             Variable      Standardized
                           Communalities
                 1             0.5495
                 2             0.5729
                 3             0.3835
                 4             0.7877
                 5             0.6195
                 6             0.8231
                 7             0.6005
                 8             0.5384
                 9             0.7691

  Eigenvalues

   0.1597E+02  0.4358E+01  0.1847E+01  0.1156E+01  0.1119E+01  0.1027E+01
   0.9257E+00  0.8951E+00  0.8771E+00

    Test Statistic =  7.149
                df = 12.000
 Significance level =  0.848

  Residuals

    0.000
   -0.013   0.022
    0.011  -0.005   0.023
   -0.010  -0.019  -0.016   0.003
   -0.005   0.011  -0.012  -0.001  -0.001
    0.015  -0.022  -0.011   0.002   0.029  -0.012
   -0.001  -0.011   0.013   0.005  -0.006  -0.001   0.003
   -0.006   0.010  -0.005  -0.011   0.002   0.007   0.003  -0.001

  Loadings, Communalities and PSI

    0.664  -0.321   0.074   0.550   0.450
    0.689  -0.247  -0.193   0.573   0.427
    0.493  -0.302  -0.222   0.383   0.617
    0.837   0.292  -0.035   0.788   0.212
    0.705   0.315  -0.153   0.619   0.381
    0.819   0.377   0.105   0.823   0.177
    0.661  -0.396  -0.078   0.600   0.400
    0.458  -0.296   0.491   0.538   0.462
    0.766  -0.427  -0.012   0.769   0.231
```

G03CCF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03CCF computes factor score coefficients from the result of fitting a factor analysis model by maximum likelihood as performed by G03CAF.

2. Specification

```
      SUBROUTINE G03CCF (METHOD, ROTATE, NVAR, NFAC, FL, LDFL, PSI,
     1                   E, R, LDR, FS, LDFS, WK, IFAIL)
      INTEGER            NVAR, NFAC, LDFL, LDR, LDFS, IFAIL
      real               FL(LDFL,NFAC), PSI(NVAR), E(NVAR), R(LDR,*),
     1                   FS(LDFS,NFAC), WK(NVAR)
      CHARACTER*1        METHOD, ROTATE
```

3. Description

A factor analysis model aims to account for the covariances among p variables, observed on n individuals, in terms of a smaller number, k, of unobserved variables or factors. The values of the factors for an individual are known as factor scores. G03CAF fits the factor analysis model by maximum likelihood and returns the estimated factor loading matrix, Λ, and the diagonal matrix of variances of the unique components, Ψ. To obtain estimates of the factors a p by k matrix of factor score coefficients, Φ, is formed. The estimated vector of factor scores, $\hat{f}$, is then given by:

$$\hat{f} = x^T \Phi,$$

where x is the vector of observed variables for an individual.

There are two commonly used methods of obtaining factor score coefficients.

The regression method:

$$\Phi = \Psi^{-1}\Lambda(I+\Lambda^T\Psi^{-1}\Lambda)^{-1},$$

and Bartlett's method:

$$\Phi = \Psi^{-1}\Lambda(\Lambda^T\Psi^{-1}\Lambda)^{-1}.$$

See Lawley and Maxwell [1] for details of both methods. In the regression method as given above, it is assumed that the factors are not correlated and have unit variance; this is true for models fitted by G03CAF. Further, for models fitted by G03CAF,

$$\Lambda^T\Psi^{-1}\Lambda = \Theta - I,$$

where Θ is the diagonal matrix of eigenvalues of the matrix S^*, as described in G03CAF.

The factors may be orthogonally rotated using an orthogonal rotation matrix, R, as computed by G03BAF. The factor scores for the rotated matrix are then given by ΛR.

4. References

[1] LAWLEY, D.N. and MAXWELL, A.E.
Factor Analysis as a Statistical Method.
Butterworths, (2nd Edition) 1971.

5. Parameters

1: METHOD – CHARACTER*1. *Input*

On entry: indicates which method is to be used to compute the factor score coefficients.

If METHOD = 'R', then the regression method is used.

If METHOD = 'B', then Bartlett's method is used.

Constraint: METHOD = 'B' or 'R'.

2: ROTATE – CHARACTER*1. *Input*

On entry: indicates whether a rotation is to be applied.

If ROTATE = 'R', then a rotation will be applied to the coefficients and the rotation matrix, R, must be given in R.

If ROTATE = 'U', then no rotation is applied.

Constraint: ROTATE = 'R' or 'U'.

3: NVAR – INTEGER. *Input*

On entry: the number of observed variables in the factor analysis, p.

Constraint: NVAR $\geq$ NFAC.

4: NFAC – INTEGER. *Input*

On entry: the number of factors in the factor analysis, k.

Constraint: NFAC ≥ 1.

5: FL(LDFL,NFAC) – ***real*** array. *Input*

On entry: the matrix of unrotated factor loadings, Λ, as returned by G03CAF.

6: LDFL – INTEGER. *Input*

On entry: the first dimension of the array FL as declared in the (sub)program from which G03CCF is called.

Constraint: LDFL $\geq$ NVAR.

7: PSI(NVAR) – ***real*** array. *Input*

On entry: the diagonal elements of Ψ, as returned by G03CAF.

Constraint: PSI(i) > 0.0, for $i = 1,2,\ldots,p$.

8: E(NVAR) – ***real*** array. *Input*

On entry: the eigenvalues of the matrix S^*, as returned by G03CAF.

Constraint: E(i) > 1.0, for $i = 1,2,\ldots,p$.

9: R(LDR,*) – ***real*** array. *Input*

Note: the second dimension of the array R must be at least 1 if ROTATE = 'U' and at least NFAC if ROTATE = 'R'.

On entry: if ROTATE = 'R', then R must contain the orthogonal rotation matrix, R, as returned by G03BAF. If ROTATE = 'U', then R need not be set.

10: LDR – INTEGER. *Input*

On entry: the first dimension of the array R as declared in the (sub)program from which G03CCF is called.

Constraint: if ROTATE = 'R', LDR $\geq$ NFAC.

11: FS(LDFS,NFAC) – ***real*** array. *Output*

On exit: the matrix of factor score coefficients, Φ. FS(i,j) contains the factor score coefficient for the jth factor and the ith observed variable, for $i = 1,2,\ldots,p$; $j = 1,2,\ldots,k$.

12: LDFS – INTEGER. *Input*

On entry: the first dimension of the array FS as declared in the (sub)program from which G03CCF is called.

Constraint: LDFS $\geq$ NVAR.

13: WK(NVAR) – ***real*** array. *Workspace*

14: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, NFAC < 1,
or NVAR < NFAC,
or LDFL < NVAR,
or LDFS < NVAR,
or ROTATE = 'R' and LDR < NFAC,
or METHOD ≠ 'R' or 'B',
or ROTATE ≠ 'R' or 'U'.

IFAIL = 2

On entry, a value of PSI ≤ 0.0,
or a value of E ≤ 1.0.

7. Accuracy

Accuracy will depend on the accuracy requested when computing the estimated factor loadings using G03CAF.

8. Further Comments

To compute the factor scores using the factor score coefficients the values for the observed variables first need to be standardized by subtracting the sample means and, if the factor analysis is based upon a correlation matrix, dividing by the sample standard deviations. This may be performed using G03ZAF. The standardized variables are then post-multiplied by the factor score coefficients. This may be performed using routines from Chapter F06, for example F06YAF.

If principal component analysis is required the routine G03AAF computes the principal component scores directly. Hence, the factor score coefficients are not needed.

9. Example

The example is taken from Lawley and Maxwell [1]. The correlation matrix for 220 observations on six school subjects is input and a factor analysis model with two factors fitted using G03CAF. The factor score coefficients are computed using the regression method.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03CCF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, LWK
      PARAMETER        (NMAX=20,MMAX=10,LWK=400)
```

```
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, M, N, NFAC, NVAR
      CHARACTER        MATRIX, METHOD, WEIGHT
*     .. Local Arrays ..
      real             COM(MMAX), E(MMAX), FL(MMAX,MMAX),
FS(MMAX,MMAX),
     +                 PSI(MMAX), R(MMAX,MMAX), STAT(4), WK(LWK),
     +                 WT(NMAX), X(NMAX,MMAX)
      INTEGER          IOP(5), ISX(MMAX), IWK(4*MMAX+2)
*     .. External Subroutines ..
      EXTERNAL         G03CAF, G03CCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03CCF Example Program Results'
*     Skip headings in data file
      READ (NIN,*)
      READ (NIN,*) MATRIX, WEIGHT, N, M, NVAR, NFAC
      IF (M.LE.MMAX .AND. (MATRIX.EQ.'C' .OR. MATRIX.EQ.'c' .OR. N.LE.
     +    NMAX)) THEN
         IF (MATRIX.EQ.'C' .OR. MATRIX.EQ.'c') THEN
            DO 20 I = 1, M
               READ (NIN,*) (X(I,J),J=1,M)
   20       CONTINUE
         ELSE
            IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
               DO 40 I = 1, N
                  READ (NIN,*) (X(I,J),J=1,M), WT(I)
   40          CONTINUE
            ELSE
               DO 60 I = 1, N
                  READ (NIN,*) (X(I,J),J=1,M)
   60          CONTINUE
            END IF
         END IF
         READ (NIN,*) (ISX(J),J=1,M)
         READ (NIN,*) (IOP(J),J=1,5)
         IFAIL = 1
*
         CALL G03CAF(MATRIX,WEIGHT,N,M,X,NMAX,NVAR,ISX,NFAC,WT,E,STAT,
     +               COM,PSI,R,FL,MMAX,IOP,IWK,WK,LWK,IFAIL)
*
         IF (IFAIL.EQ.0 .OR. IFAIL.GT.4) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Loadings, Communalities and PSI'
            WRITE (NOUT,*)
            DO 80 I = 1, NVAR
               WRITE (NOUT,99999) (FL(I,J),J=1,NFAC), COM(I), PSI(I)
   80       CONTINUE
            READ (NIN,*) METHOD
            IFAIL = 0
*
            CALL G03CCF(METHOD,'U',NVAR,NFAC,FL,MMAX,PSI,E,R,MMAX,FS,
     +                  MMAX,WK,IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Factor score coefficients'
            WRITE (NOUT,*)
            DO 100 I = 1, NVAR
               WRITE (NOUT,99999) (FS(I,J),J=1,NFAC)
  100       CONTINUE
         END IF
      END IF
      STOP
*
99999 FORMAT (2X,4F8.3)
      END
```

9.2. Program Data

```
G03CCF Example Program Data
'C' 'U' 220 6  6 2
1.000 0.439 0.410 0.288 0.329 0.248
0.439 1.000 0.351 0.354 0.320 0.329
0.410 0.351 1.000 0.164 0.190 0.181
0.288 0.354 0.164 1.000 0.595 0.470
0.329 0.320 0.190 0.595 1.000 0.464
0.248 0.329 0.181 0.470 0.464 1.000
  1    1    1   1    1   1
1 -1 500 3 5
'R'
```

9.3. Program Results

```
 G03CCF Example Program Results

  Loadings, Communalities and PSI

     0.553  -0.429   0.490   0.510
     0.568  -0.288   0.406   0.594
     0.392  -0.450   0.356   0.644
     0.740   0.273   0.623   0.377
     0.724   0.211   0.569   0.431
     0.595   0.132   0.372   0.628

  Factor score coefficients

     0.193  -0.392
     0.170  -0.226
     0.109  -0.326
     0.349   0.337
     0.299   0.229
     0.169   0.098
```

G03DAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03DAF computes a test statistic for the equality of within-group covariance matrices and also computes matrices for use in discriminant analysis.

2. Specification

```
SUBROUTINE G03DAF (WEIGHT, N, M, X, LDX, ISX, NVAR, ING, NG,
1                  WT, NIG, GMEAN, LDG, DET, GC, STAT, DF,
2                  SIG, WK, IWK, IFAIL)
INTEGER          N, M, LDX, ISX(M), NVAR, ING(N), NG, NIG(NG), LDG,
1                IWK(NG), IFAIL
real             X(LDX,M), WT(*), GMEAN(LDG,NVAR), DET(NG),
1                GC((NG+1)*NVAR*(NVAR+1)/2), STAT, DF, SIG,
2                WK(N*(NVAR+1))
CHARACTER*1      WEIGHT
```

3. Description

Let a sample of n observations on p variables come from n_g groups with n_j observations in the jth group and $\sum n_j = n$. If the data is assumed to follow a multivariate Normal distribution with the variance-covariance matrix of the jth group Σ_j, then to test for equality of the variance-covariance matrices between groups, that is $\Sigma_1 = \Sigma_2 = \ldots = \Sigma_{n_g} = \Sigma$, the following likelihood-ratio test statistic, G, can be used;

$$G = C\left\{(n-n_g)\log|S|-\sum_{j=1}^{n_g}(n_j-1)\log|S_j|\right\},$$

where

$$C = 1 - \frac{2p^2+3p-1}{6(p+1)(n_g-1)}\left(\sum_{j=1}^{n_g}\frac{1}{(n_j-1)} - \frac{1}{(n-n_g)}\right),$$

and S_j are the within-group variance-covariance matrices and S is the pooled variance-covariance matrix given by

$$S = \frac{\sum_{j=1}^{n_g}(n_j-1)S_j}{(n-n_g)}.$$

For large n, G is approximately distributed as a χ^2 variable with $\frac{1}{2}p(p+1)(n_g-1)$ degrees of freedom, see Morrison [4] for further comments. If weights are used, then S and S_j are the weighted pooled and within-group variance-covariance matrices and n is the effective number of observations, that is the sum of the weights.

Instead of calculating the within group variance-covariance matrices and then computing their determinants in order to calculate the test statistic, G03DAF uses a QR decomposition. The group means are subtracted from the data and then for each group a QR decomposition is computed to give an upper triangular matrix R_j^*. This matrix can be scaled to give a matrix R_j such that $S_j = R_j^T R_j$. The pooled R matrix is then computed from the R_j matrices. The values of $|S|$ and the $|S_j|$ can then be calculated from the diagonal elements of R and the R_j.

This approach means that the Mahalanobis squared distances for a vector observation x can be computed as $z^T z$, where $R_j z = (x-\bar{x}_j)$, $\bar{x}_j$ being the vector of means of the jth group. These distances can be calculated by G03DBF. The distances are used in discriminant analysis and G03DCF uses the results of G03DAF to perform several different types of discriminant analysis.

The differences between the discriminant methods are, in part, due to whether or not the within-group variance-covariance matrices are equal.

4. References

[1] AITCHISON, J. and DUNSMORE, I.R.
Statistical Prediction Analysis.
Cambridge, 1975.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics.
Griffin, Vol. 3, 1976.

[3] KRZANOWSKI, W.J.
Principles of Multivariate Analysis.
Oxford University Press, 1990.

[4] MORRISON, D.F.
Multivariate Statistical Methods.
McGraw-Hill, 1967.

5. Parameters

1: WEIGHT – CHARACTER*1. *Input*

On entry: indicates if weights are to be used.

If WEIGHT = 'U' (Unweighted), no weights are used.

If WEIGHT = 'W' (Weighted), weights are to be used and must be supplied in WT.

Constraint: WEIGHT = 'U' or 'W'.

2: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N $\geq$ 1.

3: M – INTEGER. *Input*

On entry: the number of variables in the data array X.

Constraint: M $\geq$ NVAR.

4: X(LDX,M) – ***real*** array. *Input*

On entry: X(k,l) must contain the kth observation for the lth variable, for $k = 1,2,...,n$; l = 1,2,...,M.

5: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03DAF is called.

Constraint: LDX $\geq$ N.

6: ISX(M) – INTEGER array. *Input*

On entry: ISX(l) indicates whether or not the lth variable in X is to be included in the variance-covariance matrices.

If ISX(l) > 0 the lth variable is included, for l = 1,2,...,M; otherwise it is not referenced.

Constraint: ISX(l) > 0 for NVAR values of l.

7: NVAR – INTEGER. *Input*

On entry: the number of variables in the variance-covariance matrices, p.

Constraint: NVAR $\geq$ 1.

8: ING(N) – INTEGER array. *Input*

On entry: ING(k) indicates to which group the kth observation belongs, for $k = 1,2,...,n$.

Constraint: $1 \leq$ ING$(k) \leq$ NG for $k = 1,2,...,n$ and the values of ING must be such that each group has at least NVAR members.

9: NG – INTEGER. *Input*

On entry: the number of groups, n_g.

Constraint: NG ≥ 2.

10: WT(*) – ***real*** array. *Input*

On entry: if WEIGHT = 'W' the first n elements of WT must contain the weights to be used in the analysis and the effective number of observations for a group is the sum of the weights of the observations in that group. If WT(k)=0.0 the kth observation is excluded from the calculations.

If WEIGHT = 'U', WT is not referenced and the effective number of observations for a group is the number of observations in that group.

Constraint: if WEIGHT = 'W', WT$(k) \geq 0.0$ for $k = 1,2,...,n$ and the effective number of observations for each group must be greater than 1.

11: NIG(NG) – INTEGER array. *Output*

On exit: NIG(j) contains the number of observations in the jth group, for $j = 1,2,...,n_g$.

12: GMEAN(LDG,NVAR) – ***real*** array. *Output*

On exit: the jth row of GMEAN contains the means of the p selected variables for the jth group, for $j = 1,2,...,n_g$.

13: LDG – INTEGER. *Input*

On entry: the first dimension of the array GMEAN as declared in the (sub)program from which G03DAF is called.

Constraint: LDG $\geq$ NG.

14: DET(NG) – ***real*** array. *Output*

On exit: the logarithm of the determinants of the within-group variance-covariance matrices.

15: GC((NG+1)*NVAR*(NVAR+1)/2) – ***real*** array. *Output*

On exit: the first $p(p+1)/2$ elements of GC contain R and the remaining n_g blocks of $p(p+1)/2$ elements contain the R_j matrices. All are stored in packed form by columns.

16: STAT – ***real.*** *Output*

On exit: the likelihood-ratio test statistic, G.

17: DF – ***real.*** *Output*

On exit: the degrees of freedom for the distribution of G.

18: SIG – ***real.*** *Output*

On exit: the significance level for G.

19: WK(N*(NVAR+1)) – ***real*** array. *Workspace*

20: IWK(NG) – INTEGER array. *Workspace*

21: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, NVAR < 1,
or N < 1,
or NG < 2,
or M < NVAR,
or LDX < N,
or LDG < NG,
or WEIGHT ≠ 'U' or 'W'.

IFAIL = 2

On entry, WEIGHT = 'W' and a value of WT < 0.0.

IFAIL = 3

On entry, there are not exactly NVAR elements of ISX > 0,
or a value of ING is not in the range 1 to NG,
or the effective number of observations for a group is less than 1,
or a group has less than NVAR members.

IFAIL = 4

R or one of the R_j is not of full rank.

7. Accuracy

The accuracy is dependent on the accuracy of the computation of the QR decomposition. See F01QCF for further details.

8. Further Comments

The time will be approximately proportional to np^2.

9. Example

The data, taken from Aitchison and Dunsmore [1], is concerned with the diagnosis of three 'types' of Cushing's syndrome. The variables are the logarithms of the urinary excretion rates (mg/24hr) of two steroid metabolites. Observations for a total of 21 patients are input and the statistics computed by G03DAF. The printed results show that there is evidence that the within-group variance-covariance matrices are not equal.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03DAF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, GPMAX
      PARAMETER        (NMAX=21,MMAX=2,GPMAX=3)
*     .. Local Scalars ..
      real             DF, SIG, STAT
      INTEGER          I, IFAIL, J, M, N, NG, NVAR
      CHARACTER        WEIGHT
*     .. Local Arrays ..
      real             DET(GPMAX), GC((GPMAX+1)*MMAX*(MMAX+1)/2),
     +                 GMEAN(GPMAX,MMAX), WK(NMAX*(MMAX+1)), WT(NMAX),
     +                 X(NMAX,MMAX)
      INTEGER          ING(NMAX), ISX(MMAX), IWK(GPMAX), NIG(GPMAX)
*     .. External Subroutines ..
      EXTERNAL         G03DAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03DAF Example Program Results'
*     Skip headings in data file
      READ (NIN,*)
      READ (NIN,*) N, M, NVAR, NG, WEIGHT
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
            DO 20 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), ING(I), WT(I)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), ING(I)
   40       CONTINUE
         END IF
         READ (NIN,*) (ISX(J),J=1,M)
         IFAIL = 0
*
         CALL G03DAF(WEIGHT,N,M,X,NMAX,ISX,NVAR,ING,NG,WT,NIG,GMEAN,
     +               GPMAX,DET,GC,STAT,DF,SIG,WK,IWK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Group means'
         WRITE (NOUT,*)
         DO 60 I = 1, NG
            WRITE (NOUT,99999) (GMEAN(I,J),J=1,NVAR)
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' LOG of determinants'
         WRITE (NOUT,*)
         WRITE (NOUT,99999) (DET(J),J=1,NG)
         WRITE (NOUT,*)
         WRITE (NOUT,99998) ' STAT = ', STAT
         WRITE (NOUT,99998) '   DF = ', DF
         WRITE (NOUT,99998) '  SIG = ', SIG
      END IF
      STOP
*
99999 FORMAT (1X,3F10.4)
99998 FORMAT (1X,A,F7.4)
      END
```

9.2. Program Data

```
G03DAF Example Program Data
  21 2 2 3 'U'
  1.1314    2.4596   1
  1.0986    0.2624   1
  0.6419   -2.3026   1
  1.3350   -3.2189   1
  1.4110    0.0953   1
  0.6419   -0.9163   1
  2.1163    0.0000   2
  1.3350   -1.6094   2
  1.3610   -0.5108   2
  2.0541    0.1823   2
  2.2083   -0.5108   2
  2.7344    1.2809   2
  2.0412    0.4700   2
  1.8718   -0.9163   2
  1.7405   -0.9163   2
  2.6101    0.4700   2
  2.3224    1.8563   3
  2.2192    2.0669   3
  2.2618    1.1314   3
  3.9853    0.9163   3
  2.7600    2.0281   3
  1     1
```

9.3. Program Results

```
G03DAF Example Program Results

 Group means

    1.0433   -0.6034
    2.0073   -0.2060
    2.7097    1.5998

 LOG of determinants

   -0.8273   -3.0460   -2.2877

 STAT = 19.2410
   DF =  6.0000
  SIG =  0.0038
```

G03DBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03DBF computes Mahalanobis squared distances for group or pooled variance-covariance matrices. It is intended for use after G03DAF.

2. Specification

```
      SUBROUTINE G03DBF (EQUAL, MODE, NVAR, NG, GMEAN, LDG, GC,
     1                   NOBS, M, ISX, X, LDX, D, LDD, WK, IFAIL)
      INTEGER          NVAR, NG, LDG, NOBS, M, ISX(*), LDX, LDD, IFAIL
      real             GMEAN(LDG,NVAR), GC((NG+1)*NVAR*(NVAR+1)/2),
     1                 X(LDX,*), D(LDD,NG), WK(2*NVAR)
      CHARACTER*1      EQUAL, MODE
```

3. Description

Consider p variables observed on n_g populations or groups. Let $\bar{x}_j$ be the sample mean and S_j the within-group variance-covariance matrix for the jth group and let x_k be the kth sample point in a data set. A measure of the distance of the point from the jth population or group is given by the Mahalanobis distance, D_{kj}^2:

$$D_{kj}^2 = (x_k - \bar{x}_j)^T S_j^{-1} (x_k - \bar{x}_j).$$

If the pooled estimated of the variance-covariance matrix S is used rather than the within-group variance-covariance matrices, then the distance is:

$$D_{kj}^2 = (x_k - \bar{x}_j)^T S^{-1} (x_k - \bar{x}_j).$$

Instead of using the variance-covariance matrices S and S_j, G03DBF uses the upper triangular matrices R and R_j supplied by G03DAF such that $S = R^T R$ and $S_j = R_j^T R_j$. D_{kj}^2 can then be calculated as $z^T z$ where $R_j z = (x_k - \bar{x}_j)$ or $Rz = (x_k - \bar{x}_j)$ as appropriate.

A particular case is when the distance between the group or population means is to be estimated. The Mahalanobis distance between the ith and jth groups is:

$$D_{ij}^2 = (\bar{x}_i - \bar{x}_j)^T S_j^{-1} (\bar{x}_i - \bar{x}_j)$$

or

$$D_{ij}^2 = (\bar{x}_i - \bar{x}_j)^T S^{-1} (\bar{x}_i - \bar{x}_j).$$

Note: $D_{jj}^2 = 0$ and that in the case when the pooled variance-covariance matrix is used $D_{ij}^2 = D_{ji}^2$ so in this case only the lower triangular values of D_{ij}^2, $i > j$, are computed.

4. References

[1] AITCHISON, J. and DUNSMORE, I.R.
Statistical Prediction Analysis.
Cambridge, 1975.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics.
Griffin, Vol. 3, 1976.

[3] KRZANOWSKI, W.J.
Principles of Multivariate Analysis.
Oxford University Press, 1990.

5. Parameters

1: EQUAL – CHARACTER*1. *Input*

On entry: indicates whether or not the within-group variance-covariance matrices are assumed to be equal and the pooled variance-covariance matrix used.

If EQUAL = 'E' the within-group variance-covariance matrices are assumed equal and the matrix R stored in the first $p(p+1)/2$ elements of GC is used.

If EQUAL = 'U' the within-group variance-covariance matrices are assumed to be unequal and the matrices R_j, for $j = 1,2,...,n_g$, stored in the remainder of GC are used.

Constraint: EQUAL = 'E' or 'U'.

2: MODE – CHARACTER*1. *Input*

On entry: indicates whether distances from sample points are to be calculated or distances between the group means.

If MODE = 'S' the distances between the sample points given in X and the group means are calculated.

If MODE = 'M' the distances between the group means will be calculated.

Constraint: MODE = 'M' or 'S'.

3: NVAR – INTEGER. *Input*

On entry: the number of variables, p, in the variance-covariance matrices as specified to G03DAF.

Constraint: NVAR $\geq$ 1.

4: NG – INTEGER. *Input*

On entry: the number of groups, n_g.

Constraint: NG $\geq$ 2.

5: GMEAN(LDG,NVAR) – ***real*** array. *Input*

On entry: the jth row of GMEAN contains the means of the p selected variables for the jth group, for $j = 1,2,...,n_g$. These are returned by G03DAF.

6: LDG – INTEGER. *Input*

On entry: the first dimension of the array GMEAN as declared in the (sub)program from which G03DBF is called.

Constraint: LDG $\geq$ NG.

7: GC((NG+1)*NVAR*(NVAR+1)/2) – ***real*** array. *Input*

On entry: the first $p(p+1)/2$ elements of GC should contain the upper triangular matrix R and the next n_g blocks of $p(p+1)/2$ elements should contain the upper triangular matrices R_j. All matrices must be stored packed by column. These matrices are returned by G03DAF. If EQUAL = 'E' only the first $p(p+1)/2$ elements are referenced, if EQUAL = 'U' only the elements $p(p+1)/2 + 1$ to $(n_g+1)p(p+1)/2$ are referenced.

Constraints: if EQUAL = 'E' the diagonal elements of $R \neq 0.0$,
if EQUAL = 'U' the diagonal elements of the $R_j \neq 0.0$, for $j = 1,2,...,$NG.

8: NOBS – INTEGER. *Input*

On entry: if MODE = 'S' the number of sample points in X for which distances are to be calculated. If MODE = 'M', NOBS is not referenced.

Constraint: if MODE = 'S', NOBS $\geq$ 1.

9: M – INTEGER. *Input*

On entry: if MODE = 'S' the number of variables in the data array X. If MODE = 'M', then M is not referenced.

Constraint: if MODE = 'S', M $\geq$ NVAR.

10: ISX(*) – INTEGER array. *Input*

Note: the dimension of the array ISX must be at least max(1,M).

On entry: if MODE = 'S', ISX(l) indicates if the lth variable in X is to be included in the distance calculations. If ISX(l) > 0 the lth variable is included, for l = 1,2,...,M; otherwise the lth variable is not referenced.

If MODE = 'M', then ISX is not referenced.

Constraint: if MODE = 'S', ISX(l) > 0 for NVAR values of l.

11: X(LDX,*) – ***real*** array. *Input*

Note: the second dimension of the array X must be at least max(1,M).

On entry: if MODE = 'S' the kth row of X must contain x_k. That is X(k,l) must contain the kth sample value for the lth variable for k = 1,2,...,NOBS; l = 1,2,...,M. Otherwise X is not referenced.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03DBF is called.

Constraint: if MODE = 'S', LDX $\geq$ NOBS.

13: D(LDD,NG) – ***real*** array. *Output*

On exit: the squared distances.

If MODE = 'S', D(k,j) contains the squared distance of the kth sample point from the jth group mean, D_{kj}^2, for k = 1,2,...,NOBS; $j = 1,2,\ldots,n_g$.

If MODE = 'M' and EQUAL = 'U', D(i,j) contains the squared distance between the ith mean and the jth mean, D_{ij}^2, for $i = 1,2,\ldots,n_g$; $j = 1,2,\ldots,i-1,i+1,\ldots,n_g$. The elements D($i,i$) are not referenced for $i = 1,2,\ldots,n_g$.

If MODE = 'M' and EQUAL = 'E', D(i,j) contains the squared distance between the ith mean and the jth mean, D_{ij}^2, for $i = 1,2,\ldots,n_g$; $j = 1,2,\ldots,i-1$. Since $D_{ij} = D_{ji}$ the elements D(i,j) are not referenced, for $i = 1,2,\ldots,n_g$; $j = i,i+1,\ldots,n_g$.

14: LDD – INTEGER. *Input*

On entry: the first dimension of the array D as declared in the (sub)program from which G03DBF is called.

Constraint: if MODE = 'S', LDD $\geq$ NOBS; if MODE = 'M', LDD $\geq$ NG.

15: WK(2*NVAR) – ***real*** array. *Workspace*

16: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, NVAR < 1,
or NG < 2,
or LDG < NG,
or MODE = 'S' and NOBS < 1,
or MODE = 'S' and M < NVAR,
or MODE = 'S' and LDX < NOBS,
or MODE = 'S' and LDD < NOBS,
or MODE = 'M' and LDD < NG,
or EQUAL ≠ 'E' or 'U',
or MODE ≠ 'M' or 'S'.

IFAIL = 2

On entry, MODE = 'S' and the number of variables indicated by ISX is not equal to NVAR,
or EQUAL = 'E' and a diagonal element of R is zero,
or EQUAL = 'U' and a diagonal element of R_j for some j is zero.

7. Accuracy

The accuracy will depend upon the accuracy of the input R or R_j matrices.

8. Further Comments

If the distances are to be used for discrimination, see also G03DCF.

9. Example

The data, taken from Aitchison and Dunsmore [1], is concerned with the diagnosis of three 'types' of Cushing's syndrome. The variables are the logarithms of the urinary excretion rates (mg/24hr) of two steroid metabolites. Observations for a total of 21 patients are input and the group means and R matrices are computed by G03DAF. A further six observations of unknown type are input, and the distances from the group means of the 21 patients of known type are computed under the assumption that the within-group variance-covariance matrices are not equal. These results are printed and indicate that the first four are close to one of the groups while observations 5 and 6 are some distance from any group.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03DBF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, GPMAX
      PARAMETER        (NMAX=21,MMAX=2,GPMAX=3)
*     .. Local Scalars ..
      real             DF, SIG, STAT
      INTEGER          I, IFAIL, J, M, N, NG, NOBS, NVAR
      CHARACTER        EQUAL, WEIGHT
```

```
*     .. Local Arrays ..
      real              D(NMAX,GPMAX), DET(GPMAX),
     +                  GC((GPMAX+1)*MMAX*(MMAX+1)/2), GMEAN(GPMAX,MMAX),
     +                  WK(NMAX*(MMAX+1)), WT(NMAX), X(NMAX,MMAX)
      INTEGER           ING(NMAX), ISX(MMAX), IWK(GPMAX), NIG(GPMAX)
*     .. External Subroutines ..
      EXTERNAL          G03DAF, G03DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03DBF Example Program Results'
*     Skip headings in data file
      READ (NIN,*)
      READ (NIN,*) N, M, NVAR, NG, WEIGHT
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
            DO 20 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), ING(I), WT(I)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), ING(I)
   40       CONTINUE
         END IF
         READ (NIN,*) (ISX(J),J=1,M)
         IFAIL = 0
*
         CALL G03DAF(WEIGHT,N,M,X,NMAX,ISX,NVAR,ING,NG,WT,NIG,GMEAN,
     +               GPMAX,DET,GC,STAT,DF,SIG,WK,IWK,IFAIL)
*
         READ (NIN,*) NOBS, EQUAL
         IF (NOBS.LE.NMAX) THEN
            DO 60 I = 1, NOBS
               READ (NIN,*) (X(I,J),J=1,M)
   60       CONTINUE
            IFAIL = 0
*
            CALL G03DBF(EQUAL,'Sample points',NVAR,NG,GMEAN,GPMAX,GC,
     +                  NOBS,M,ISX,X,NMAX,D,NMAX,WK,IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) '  Obs          Distances'
            WRITE (NOUT,*)
            DO 80 I = 1, NOBS
               WRITE (NOUT,99999) I, (D(I,J),J=1,NG)
   80       CONTINUE
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,I3,3F10.3)
      END
```

9.2. Program Data

```
G03DBF Example Program Data
  21 2 2 3 'U'
  1.1314   2.4596   1
  1.0986   0.2624   1
  0.6419  -2.3026   1
  1.3350  -3.2189   1
  1.4110   0.0953   1
  0.6419  -0.9163   1
  2.1163   0.0000   2
  1.3350  -1.6094   2
  1.3610  -0.5108   2
  2.0541   0.1823   2
  2.2083  -0.5108   2
  2.7344   1.2809   2
  2.0412   0.4700   2
  1.8718  -0.9163   2
  1.7405  -0.9163   2
```

```
2.6101     0.4700     2
2.3224     1.8563     3
2.2192     2.0669     3
2.2618     1.1314     3
3.9853     0.9163     3
2.7600     2.0281     3
 1          1
 6 'U'
1.6292    -0.9163
2.5572     1.6094
2.5649    -0.2231
0.9555    -2.3026
3.4012    -2.3026
3.0204    -0.2231
```

9.3. Program Results

```
G03DBF Example Program Results

  Obs            Distances

  1      3.339       0.752      50.928
  2     20.777       5.656       0.060
  3     21.363       4.841      19.498
  4      0.718       6.280     124.732
  5     55.000      88.860      71.785
  6     36.170      15.785      15.749
```

G03DCF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03DCF allocates observations to groups according to selected rules. It is intended for use after G03DAF.

2. Specification

```
      SUBROUTINE G03DCF (TYPE, EQUAL, PRIORS, NVAR, NG, NIG, GMEAN,
     1                   LDG, GC, DET, NOBS, M, ISX, X, LDX, PRIOR,
     2                   P, LDP, IAG, ATIQ, ATI, WK, IFAIL)
      INTEGER          NVAR, NG, NIG(NG), LDG, NOBS, M, ISX(M), LDX, LDP,
     1                 IAG(NOBS), IFAIL
      real             GMEAN(LDG,NVAR), GC((NG+1)*NVAR*(NVAR+1)/2),
     1                 DET(NG), X(LDX,M), PRIOR(NG), P(LDP,NG),
     2                 ATI(LDP,*), WK(2*NVAR)
      LOGICAL          ATIQ
      CHARACTER*1      TYPE, EQUAL, PRIORS
```

3. Description

Discriminant analysis is concerned with the allocation of observations to groups using information from other observations whose group membership is known, X_t; these are called the training set. Consider p variables observed on n_g populations or groups. Let $\bar{x}_j$ be the sample mean and S_j the within-group variance-covariance matrix for the jth group; these are calculated from a training set of n observations with n_j observations in the jth group, and let x_k be the kth observation from the set of observations to be allocated to the n_g groups. The observation can be allocated to a group according to a selected rule. The allocation rule or discriminant function will be based on the distance of the observation from an estimate of the location of the groups, usually the group means. A measure of the distance of the observation from the jth group mean is given by the Mahalanobis distance, D_{kj}^2:

$$D_{kj}^2 = (x_k-\bar{x}_j)^T S_j^{-1}(x_k-\bar{x}_j). \tag{1}$$

If the pooled estimate of the variance-covariance matrix S is used rather than the within-group variance-covariance matrices, then the distance is:

$$D_{kj}^2 = (x_k-\bar{x}_j)^T S^{-1}(x_k-\bar{x}_j). \tag{2}$$

Instead of using the variance-covariance matrices S and S_j, G03DCF uses the upper triangular matrices R and R_j supplied by G03DAF such that $S = R^TR$ and $S_j = R_j^TR_j$. D_{kj}^2 can then be calculated as z^Tz where $R_jz = (x_k-\bar{x}_j)$ or $Rz = (x_k-\bar{x}_j)$ as appropriate.

In addition to the distances a set of prior probabilities of group membership, π_j, for $j = 1,2,\ldots,n_g$, may be used, with $\sum\pi_j = 1$. The prior probabilities reflect the users view as to the likelihood of the observations coming from the different groups. Two common cases for prior probabilities are $\pi_1 = \pi_2 = \ldots = \pi_{n_g}$, that is equal prior probabilities, and $\pi_j = n_j/n$, for $j = 1,2,\ldots,n_g$, that is prior probabilities proportional to the number of observations in the groups in the training set.

G03DCF uses one of four allocation rules. In all four rules the p variables are assumed to follow a multivariate Normal distribution with mean μ_j and variance-covariance matrix Σ_j if the observation comes from the jth group. The different rules depend on whether or not the within-group variance-covariance matrices are assumed equal, i.e. $\Sigma_1 = \Sigma_2 = \ldots = \Sigma_{n_g}$, and whether a predictive or estimative approach is used. If $p(x_k|\mu_j,\Sigma_j)$ is the probability of observing the observation x_k from group j, then the posterior probability of belonging to group j is:

$$p(j|x_k,\mu_j,\Sigma_j) \propto p(x_k|\mu_j,\Sigma_j)\pi_j. \tag{3}$$

In the estimative approach the parameters μ_j and Σ_j in (3) are replaced by their estimates calculated from X_t. In the predictive approach a non-informative prior distribution is used for the parameters and a posterior distribution for the parameters, $p(\mu_j,\Sigma_j|X_t)$, is found. A predictive distribution is then obtained by integrating $p(j|x_k,\mu_j,\Sigma_j)p(\mu_j,\Sigma_j|X)$ over the parameter space. This predictive distribution then replaces $p(x_k|\mu_j,\Sigma_j)$ in (3). See Aitchison and Dunsmore [1], Aitchison *et al.* [2] and Moran and Murphy [5] for further details.

The observation is allocated to the group with the highest posterior probability. Denoting the posterior probabilities, $p(j|x_k,\mu_j,\Sigma_j)$, by q_j, the four allocation rules are:

(i) Estimative with equal variance-covariance matrices – Linear Discrimination.

$$\log q_j \propto -\tfrac{1}{2}D_{kj}^2 + \log\pi_j$$

(ii) Estimative with unequal variance-covariance matrices – Quadratic Discrimination.

$$\log q_j \propto -\tfrac{1}{2}D_{kj}^2 + \log\pi_j - \tfrac{1}{2}\log|S_j|$$

(iii) Predictive with equal variance-covariance matrices

$$q_j^{-1} \propto ((n_j+1)/n_j)^{p/2}\{1+[n_j/((n-n_g)(n_j+1))]D_{kj}^2\}^{(n+1-n_g)/2}$$

(iv) Predictive with unequal variance-covariance matrices

$$q_j^{-1} \propto C\{((n_j^2-1)/n_j)|S_j|\}^{p/2}\{1+(n_j/(n_j^2-1))D_{kj}^2\}^{n_j/2} \text{ where } C = \frac{\Gamma(\frac{1}{2}(n_j-p))}{\Gamma(\frac{1}{2}n_j)}$$

In the above the appropriate value of D_{kj}^2 from (1) or (2) is used. The values of the q_j are standardized so that,

$$\sum_{j=1}^{n_g} q_j = 1.$$

Moran and Murphy [5] show the similarity between the predictive methods and methods based upon likelihood ratio tests.

In addition to allocating the observation to a group G03DCF computes an atypicality index, $I_j(x_k)$. This represents the probability of obtaining an observation more typical of group j than the observed x_k, see Aitchison and Dunsmore [1] and Aitchison *et al.* [2]. The atypicality index is computed as:

$$I_j(x_k) = P(B\le z : \tfrac{1}{2}p,\tfrac{1}{2}(n_j-d))$$

where $P(B\le\beta : a,b)$ is the lower tail probability from a beta distribution where for unequal within-group variance-covariance matrices,

$$z = D_{kj}^2/(D_{kj}^2+(n_j^2-1)/n_j),$$

and for equal within-group variance-covariance matrices,

$$z = D_{kj}^2/(D_{kj}^2+(n-n_g)(n_j-1)/n_j).$$

If $I_j(x_k)$ is close to 1 for all groups it indicates that the observation may come from a grouping not represented in the training set. Moran and Murphy [5] provide a frequentist interpretation of $I_j(x_k)$.

4. References

[1] AITCHISON, J. and DUNSMORE, I.R.
Statistical Prediction Analysis.
Cambridge, 1975.

[2] AITCHISON, J., HABBEMA, J.D.F. and KAY, J.W.
A Critical Comparison of Two Methods of Statistical Discrimination.
Appl. Statist., 26, pp. 15-25, 1977.

[3] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics.
Griffin, Vol. 3, 1976.

[4] KRZANOWSKI, W.J.
Principles of Multivariate Analysis.
Oxford University Press, 1990.

[5] MORAN, M.A. and MURPHY, B.J.
A Closer Look at Two Alternative Methods of Statistical Discrimination.
Appl. Statist., 28, pp. 223-232, 1979.

[6] MORRISON, D.F.
Multivariate Statistical Methods.
McGraw-Hill, 1967.

5. Parameters

1: TYPE – CHARACTER*1. *Input*

On entry: whether the estimative or predictive approach is used.

If TYPE = 'E' the estimative approach is used.

If TYPE = 'P' the predictive approach is used.

Constraint: TYPE = 'E' or 'P'.

2: EQUAL – CHARACTER*1. *Input*

On entry: indicates whether or not the within-group variance-covariance matrices are assumed to be equal and the pooled variance-covariance matrix used.

If EQUAL = 'E' the within-group variance-covariance matrices are assumed equal and the matrix R stored in the first $p(p+1)/2$ elements of GC is used.

If EQUAL = 'U' the within-group variance-covariance matrices are assumed to be unequal and the matrices R_i, for $i = 1,2,\ldots,n_g$, stored in the remainder of GC are used.

Constraint: EQUAL = 'E' or 'U'.

3: PRIORS – CHARACTER*1. *Input*

On entry: indicates the form of the prior probabilities to be used.

If PRIORS = 'E', equal prior probabilities are used.

If PRIORS = 'P', prior probabilities proportional to the group sizes in the training set, n_j, are used.

If PRIORS = 'I', the prior probabilities are input in PRIOR.

Constraint: PRIORS = 'E', 'I' or 'P'.

4: NVAR – INTEGER. *Input*

On entry: the number of variables, p, in the variance-covariance matrices.

Constraint: NVAR $\geq$ 1.

5: NG – INTEGER. *Input*

On entry: the number of groups, n_g.

Constraint: NG $\geq$ 2.

6: NIG(NG) – INTEGER array. *Input*

On entry: the number of observations in each group in the training set, n_j.

Constraints: if EQUAL = 'E', NIG(j) > 0, for $j = 1,2,\ldots,n_g$ and

$$\sum_{j=1}^{n_g} \mathrm{NIG}(j) > \mathrm{NG} + \mathrm{NVAR}.$$

If EQUAL = 'U', NIG(j) > NVAR, for $j = 1,2,\ldots,n_g$.

7: GMEAN(LDG,NVAR) – ***real*** array. *Input*

On entry: the jth row of GMEAN contains the means of the p variables for the jth group, for $j = 1,2,...,n_j$. These are returned by G03DAF.

8: LDG – INTEGER. *Input*

On entry: the first dimension of the array GMEAN as declared in the (sub)program from which G03DCF is called.

Constraint: LDG $\geq$ NG.

9: GC((NG+1)*NVAR*(NVAR+1)/2) – ***real*** array. *Input*

On entry: the first $p(p+1)/2$ elements of GC should contain the upper triangular matrix R and the next n_g blocks of $p(p+1)/2$ elements should contain the upper triangular matrices R_j. All matrices must be stored packed by column. These matrices are returned by G03DAF. If EQUAL = 'E' only the first $p(p+1)/2$ elements are referenced, if EQUAL = 'U' only the elements $p(p+1)/2 + 1$ to $(n_g+1)p(p+1)/2$ are referenced.

Constraints: if EQUAL = 'E' the diagonal elements of R must be $\neq$ 0.0,
if EQUAL = 'U' the diagonal elements of the R_j must be $\neq$ 0.0, for $j = 1,2,...,n_g$.

10: DET(NG) – ***real*** array. *Input*

On entry: if EQUAL = 'U' the logarithms of the determinants of the within-group variance-covariance matrices as returned by G03DAF. Otherwise DET is not referenced.

11: NOBS – INTEGER. *Input*

On entry: the number of observations in X which are to be allocated.

Constraint: NOBS $\geq$ 1.

12: M – INTEGER. *Input*

On entry: the number of variables in the data array X.

Constraint: M $\geq$ NVAR.

13: ISX(M) – INTEGER array. *Input*

On entry: ISX(l) indicates if the lth variable in X is to be included in the distance calculations.

If ISX(l) > 0 the lth variable is included, for l = 1,2,...,M; otherwise the lth variable is not referenced.

Constraint: ISX(l) > 0 for NVAR values of l.

14: X(LDX,M) – ***real*** array. *Input*

On entry: X(k,l) must contain the kth observation for the lth variable, for k = 1,2,...,NOBS; l = 1,2,...,M.

15: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03DCF is called.

Constraint: LDX $\geq$ NOBS.

16: PRIOR(NG) – ***real*** array. *Input/Output*

On entry: if PRIORS = 'I' the prior probabilities for the n_g groups.

Constraint: if PRIORS = 'I', then PRIOR(j) > 0.0 for $j = 1,2,\ldots,n_g$ and $\left|1-\sum_{j=1}^{n_g} \mathrm{PRIOR}(j)\right| \le 10\times$***machine precision***.

On exit: if PRIORS = 'P' the computed prior probabilities in proportion to group sizes for the n_g groups. If PRIORS = 'I' the input prior probabilities will be unchanged, and if PRIORS = 'E', PRIOR is not set.

17: P(LDP,NG) – ***real*** array. *Output*

On exit: P(k,j) contains the posterior probability p_{kj} for allocating the kth observation to the jth group, for $k = 1,2,\ldots$,NOBS; $j = 1,2,\ldots,n_g$.

18: LDP – INTEGER. *Input*

On entry: the first dimension of the array P as declared in the (sub)program from which G03DCF is called.

Constraint: LDP ≥ NOBS.

19: IAG(NOBS) – INTEGER array. *Output*

On exit: the groups to which the observations have been allocated.

20: ATIQ – LOGICAL. *Input*

On entry: AITQ must be .TRUE. if atypicality indices are required. If ATIQ is .FALSE. the array ATI is not set.

21: ATI(LDP,*) – ***real*** array. *Output*

Note: if ATIQ is .TRUE. the second dimension of ATI must be at least NG, if ATIQ is .FALSE. the second dimension of ATI must be at least 1.

On exit: if AITQ is .TRUE., ATI(k,j) will contain the atypicality index for the kth observation with respect to the jth group, for $k = 1,2,\ldots$,NOBS; $j = 1,2,\ldots,n_g$. If ATIQ is .FALSE., ATI is not set.

22: WK(2*NVAR) – ***real*** array. *Workspace*

23: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, NVAR < 1,
or NG < 2,
or NOBS < 1,
or M < NVAR,
or LDG < NG,
or LDX < NOBS,

or LDP < NOBS,
or TYPE ≠ 'E' or 'P',
or EQUAL ≠ 'E' or 'U',
or PRIORS ≠ 'E', 'I' or 'P'.

IFAIL = 2

On entry, the number of variables indicated by ISX is not equal to NVAR,
or EQUAL = 'E' and NIG(j) $\leq$ 0, for some j,
or EQUAL = 'E' and $\sum_{j=1}^{n_g}$ NIG(j) $\leq$ NG + NVAR,
or EQUAL = 'U' and NIG(j) $\leq$ NVAR for some j.

IFAIL = 3

On entry, PRIORS = 'I' and PRIOR(j) $\leq$ 0.0 for some j,
or PRIORS = 'I' and $\sum_{j=1}^{n_g}$ PRIOR(j) is not within 10×***machine precision*** of 1.

IFAIL = 4

On entry, EQUAL = 'E' and a diagonal element of R is zero,
or EQUAL = 'U' and a diagonal element of R_j for some j is zero.

7. Accuracy

The accuracy of the returned posterior probabilities will depend on the accuracy of the input R or R_j matrices. The atypicality index should be accurate to four significant places.

8. Further Comments

The distances D_{kj}^2 can be computed using G03DBF if other forms of discrimination are required.

9. Example

The data, taken from Aitchison and Dunsmore [1], is concerned with the diagnosis of three 'types' of Cushing's syndrome. The variables are the logarithms of the urinary excretion rates (mg/24hr) of two steroid metabolites. Observations for a total of 21 patients are input and the group means and R matrices are computed by G03DAF. A further six observations of unknown type are input and allocations made using the predictive approach and under the assumption that the within-group covariance matrices are not equal. The posterior probabilities of group membership, q_j, and the atypicality index are printed along with the allocated group. The atypicality index shows that observations 5 and 6 do not seem to be typical of the three types present in the initial 21 observations.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03DCF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, GPMAX
      PARAMETER        (NMAX=21,MMAX=2,GPMAX=3)
*     .. Local Scalars ..
      real             DF, SIG, STAT
      INTEGER          I, IFAIL, J, M, N, NG, NOBS, NVAR
      CHARACTER        EQUAL, TYPE, WEIGHT
```

```
*     .. Local Arrays ..
      real             ATI(NMAX,GPMAX), DET(GPMAX),
     +                 GC((GPMAX+1)*MMAX*(MMAX+1)/2), GMEAN(GPMAX,MMAX),
     +                 P(NMAX,GPMAX), PRIOR(GPMAX), WK(NMAX*(MMAX+1)),
     +                 WT(NMAX), X(NMAX,MMAX)
      INTEGER          IAG(NMAX), ING(NMAX), ISX(MMAX), IWK(GPMAX),
     +                 NIG(GPMAX)
*     .. External Subroutines ..
      EXTERNAL         G03DAF, G03DCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03DCF Example Program Results'
*     Skip headings in data file
      READ (NIN,*)
      READ (NIN,*) N, M, NVAR, NG, WEIGHT
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
            DO 20 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), ING(I), WT(I)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), ING(I)
   40       CONTINUE
         END IF
         READ (NIN,*) (ISX(J),J=1,M)
         IFAIL = 0
*
         CALL G03DAF(WEIGHT,N,M,X,NMAX,ISX,NVAR,ING,NG,WT,NIG,GMEAN,
     +               GPMAX,DET,GC,STAT,DF,SIG,WK,IWK,IFAIL)
*
         READ (NIN,*) NOBS, EQUAL, TYPE
         IF (NOBS.LE.NMAX) THEN
            DO 60 I = 1, NOBS
               READ (NIN,*) (X(I,J),J=1,M)
   60       CONTINUE
            IFAIL = 0
*
            CALL G03DCF(TYPE,EQUAL,'Equal priors',NVAR,NG,NIG,GMEAN,
     +                  GPMAX,GC,DET,NOBS,M,ISX,X,NMAX,PRIOR,P,NMAX,IAG,
     +                  .TRUE.,ATI,WK,IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) '   Obs      Posterior       Allocated',
     +        '    Atypicality'
            WRITE (NOUT,*)
     +        '               probabilities    to group      index'
            WRITE (NOUT,*)
            DO 80 I = 1, NOBS
               WRITE (NOUT,99999) I, (P(I,J),J=1,NG), IAG(I),
     +           (ATI(I,J),J=1,NG)
   80       CONTINUE
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,2(I6,5X,3F6.3))
      END
```

9.2. Program Data

```
G03DCF Example Program Data
  21 2 2 3 'U'
  1.1314     2.4596     1
  1.0986     0.2624     1
  0.6419    -2.3026     1
  1.3350    -3.2189     1
  1.4110     0.0953     1
  0.6419    -0.9163     1
  2.1163     0.0000     2
  1.3350    -1.6094     2
  1.3610    -0.5108     2
  2.0541     0.1823     2
  2.2083    -0.5108     2
  2.7344     1.2809     2
  2.0412     0.4700     2
  1.8718    -0.9163     2
  1.7405    -0.9163     2
  2.6101     0.4700     2
  2.3224     1.8563     3
  2.2192     2.0669     3
  2.2618     1.1314     3
  3.9853     0.9163     3
  2.7600     2.0281     3
   1         1
   6 'U' 'P'
  1.6292    -0.9163
  2.5572     1.6094
  2.5649    -0.2231
  0.9555    -2.3026
  3.4012    -2.3026
  3.0204    -0.2231
```

9.3. Program Results

```
 G03DCF Example Program Results

    Obs        Posterior         Allocated     Atypicality
               probabilities     to group      index

     1       0.094 0.905 0.002       2       0.596 0.254 0.975
     2       0.005 0.168 0.827       3       0.952 0.836 0.018
     3       0.019 0.920 0.062       2       0.954 0.797 0.912
     4       0.697 0.303 0.000       1       0.207 0.860 0.993
     5       0.317 0.013 0.670       3       0.991 1.000 0.984
     6       0.032 0.366 0.601       3       0.981 0.978 0.887
```

G03EAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03EAF computes a distance (dissimilarity) matrix.

2. Specification

```
      SUBROUTINE G03EAF (UPDATE, DIST, SCALE, N, M, X, LDX, ISX, S, D, IFAIL)
      INTEGER          N, M, LDX, ISX(M), IFAIL
      real             X(LDX,M), S(M), D(N*(N-1)/2)
      CHARACTER*1      UPDATE, DIST, SCALE
```

3. Description

Given n objects, a distance or dissimilarity matrix, is a symmetric matrix with zero diagonal elements such that the ijth element represents how far apart or how dissimilar the ith and jth objects are.

Let X be a n by p data matrix of observations of p variables on n objects then the distance between object j and object k, d_{jk}, can be defined as:

$$d_{jk} = \{\sum_{i=1}^{p} D(x_{ji}/s_i, x_{ki}/s_i)\}^{\alpha} ,$$

where x_{ji} and x_{ki} are the (j,i)th and (k,i)th elements of X, s_i is a standardization for the ith variable and $D(u,v)$ is a suitable function. Three functions are provided in G03EAF.

(a) Euclidean distance: $D(u,v) = (u-v)^2$ and $\alpha = \frac{1}{2}$.

(b) Euclidean squared distance: $D(u,v) = (u-v)^2$ and $\alpha = 1$.

(c) Absolute distance (city block metric): $D(u,v) = |u-v|$ and $\alpha = 1$.

Three standardizations are available.

(a) Standard deviation: $s_i = \sqrt{\sum_{j=1}^{n}(x_{ji}-\bar{x})^2/(n-1)}$

(b) Range: $s_i = \max(x_{1i},x_{2i},\ldots,x_{ni}) - \min(x_{1i},x_{2i},\ldots,x_{ni})$

(c) User supplied values of s_i.

In addition to the above distances there are a large number of other dissimilarity measures, particularly for dichotomous variables (see Krzanowski [2] and Everitt [1]). For the dichotomous case these measures are simple to compute and can, if suitable scaling is used, be combined with the distances computed by G03EAF using the updating option.

Dissimilarity measures for variables can be based on the correlation coefficient for continuous variables and contingency table statistics for dichotomous data, see chapters G02 and G11 repectively.

G03EAF returns the strictly lower triangle of the distance matrix.

4. References

[1] EVERITT, B.
Cluster Analysis.
Heinemann, 1974.

[2] KRZANOWSKI, W.J.
Principles of Multivariate Analysis.
Oxford University Press, 1990.

5. Parameters

1: UPDATE – CHARACTER*1. *Input*

On entry: indicates whether or not an existing matrix is to be updated.

If UPDATE = 'U' the matrix D is updated and distances are added to D.
If UPDATE = 'I' the matrix D is initialized to zero before the distances are added to D.

Constraint: UPDATE = 'U' or 'I'.

2: DIST – CHARACTER*1. *Input*

On entry: indicates which type of distances are computed.
If DIST = 'A', absolute distances.
If DIST = 'E', Euclidean distances.
If DIST = 'S', Euclidean squared distances.

Constraint: DIST = 'A', 'E' or 'S'.

3: SCALE – CHARACTER*1. *Input*

On entry: indicates the standardization of the variables to be used.
If SCALE = 'S', standard deviation.
If SCALE = 'R', range.
If SCALE = 'G', standardizations given in array S.
If SCALE = 'U', unscaled.

Constraint: SCALE = 'S', 'R', 'G' or 'U'.

4: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N $\geq$ 2.

5: M – INTEGER. *Input*

On entry: the total number of variables in array X.

Constraint: M > 0.

6: X(LDX,M) – ***real*** array. *Input*

On entry: X(i,j) must contain the value of the jth variable for the ith object, for $i = 1,2,...,n$; $j = 1,2,...,$M.

7: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03EAF is called.

Constraint: LDX $\geq$ N.

8: ISX(M) – INTEGER array. *Input*

On entry: ISX(j) indicates whether or not the jth variable in X is to be included in the distance computations.

If ISX$(j) > 0$ the jth variable is included, for $j = 1,2,...,$M; otherwise it is not referenced.

Constraint: ISX$(j) > 0$ for at least one j, $j = 1,2,...,$M.

9: S(M) – ***real*** array. *Input/Output*

On entry: if SCALE = 'G' and ISX$(j) > 0$ then S(j) must contain the scaling for variable j, for $j = 1,2,...,$M.

Constraint: if SCALE = 'G' and ISX$(j) > 0$ then S$(j) > 0.0$, for $j = 1,2,...,$M.

On exit: if SCALE = 'S' and ISX$(j) > 0$ then S(j) contains the standard deviation of the variable in the jth column of X. If SCALE = 'R' and ISX$(j) > 0$ then S(j) contains the range of the variable in the jth column of X. If SCALE = 'U' and ISX(j) > 0 then S(j) = 1.0. and if SCALE = 'G' then S is unchanged.

10: D(N*(N–1)/2) – ***real*** array. *Input/Output*

On entry: if UPDATE = 'U' then D must contain the strictly lower triangle of the distance matrix D to be updated. D must be stored packed by rows, i.e. D((i–1)(i–2)/2+j), $i > j$ must contain d_{ij}.

Constraint: if UPDATE = 'U' then D(j) ≥ 0.0, for $j = 1,2,...,n(n-1)/2$.

On exit: the strictly lower triangle of the distance matrix D stored packed by rows, i.e. d_{ij} is contained in D((i–1)(i–2)/2+j), $i > j$.

11: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, N < 2,
or LDX < N,
or M ≤ 0,
or UPDATE ≠ 'I' or 'U',
or DIST ≠ 'A', 'E' or 'S',
or SCALE ≠ 'S', 'R', 'G' or 'U'.

IFAIL = 2

On entry, ISX(j) ≤ 0 for j = 1,2,...,M,
or UPDATE = 'U' and D(j) < 0.0, for some $j = 1,2,...,n(n-1)/2$,
or SCALE = 'S' or 'R' and X(i,j) = X(i+1,j) for $i = 1,2,...,n-1$, for some j with ISX(i) > 0.
or S(j) ≤ 0.0 for some j when SCALE = 'G' and ISX(j) > 0.

7. Accuracy

The computations are believed to be stable.

8. Further Comments

G03ECF can be used to perform cluster analysis on the computed distance matrix.

9. Example

A data matrix of five observations and three variables is read in and a distance matrix is calculated from variables 2 and 3 using squared Euclidean distance with no scaling. This matrix is then printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03EAF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX
      PARAMETER        (NMAX=10,MMAX=10)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, LDX, M, N
      CHARACTER        DIST, SCALE, UPDATE
*     .. Local Arrays ..
      real             D(NMAX*(NMAX-1)/2), S(MMAX), X(NMAX,MMAX)
      INTEGER          ISX(MMAX)
*     .. External Subroutines ..
      EXTERNAL         G03EAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03EAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         READ (NIN,*) UPDATE, DIST, SCALE
         DO 20 J = 1, N
            READ (NIN,*) (X(J,I),I=1,M)
   20    CONTINUE
         READ (NIN,*) (ISX(I),I=1,M)
         READ (NIN,*) (S(I),I=1,M)
*
*        Compute the distance matrix
*
         IFAIL = 0
         LDX = NMAX
*
         CALL G03EAF(UPDATE,DIST,SCALE,N,M,X,LDX,ISX,S,D,IFAIL)
*
*        Print the distance matrix
*
         IFAIL = 0
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Distance Matrix'
         WRITE (NOUT,*)
         WRITE (NOUT,99999) '    1       2       3       4'
         WRITE (NOUT,*)
         DO 40 I = 2, N
            WRITE (NOUT,99998) I, (D(J),J=(I-1)*(I-2)/2+1,I*(I-1)/2)
   40    CONTINUE
      END IF
      STOP
*
99999 FORMAT (5X,A)
99998 FORMAT (1X,I2,2X,4(3X,F5.2))
      END
```

9.2. Program Data

```
G03EAF Example Program Data
5 3
'I' 'S' 'U'
1.0 1.0 1.0
2.0 1.0 2.0
3.0 6.0 3.0
4.0 8.0 2.0
5.0 8.0 0.0
 0   1   1
1.0 1.0 1.0
```

9.3. Program Results

```
G03EAF Example Program Results

 Distance Matrix

         1         2         3         4

 2      1.00
 3     29.00     26.00
 4     50.00     49.00      5.00
 5     50.00     53.00     13.00      4.00
```

G03ECF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03ECF performs hierarchical cluster analysis.

2. Specification

```
SUBROUTINE G03ECF (METHOD, N, D, ILC, IUC, CD, IORD, DORD, IWK, IFAIL)
INTEGER      METHOD, N, ILC(N-1), IUC(N-1), IORD(N), IWK(2*N), IFAIL
real         D(N*(N-1)/2), CD(N-1), DORD(N)
```

3. Description

Given a distance or dissimilarity matrix for n objects (see G03EAF), cluster analysis aims to group the n objects into a number of more or less homogeneous groups or clusters. With agglomerative clustering methods, a hierarchical tree is produced by starting with n clusters, each with a single object and then at each of n–1 stages, merging two clusters to form a larger cluster, until all objects are in a single cluster. This process can may be represented by a dendrogram (see G03EHF).

At each stage the clusters that are nearest are merged, methods differ as to how the distance between the new cluster and other clusters are computed. For three clusters i, j and k let n_i, n_j and n_k be the number of objects in each cluster and let d_{ij}, d_{ik} and d_{jk} be the distances between the clusters. Let clusters j and k be merged to give cluster jk, then the distance from cluster i to cluster jk, $d_{i.jk}$ can be computed in the following ways.

1. Single Link or nearest neighbour : $d_{i.jk} = min(d_{ij},d_{ik})$.
2. Complete Link or furthest neighbour : $d_{i.jk} = max(d_{ij},d_{ik})$.
3. Group average : $d_{i.jk} = \dfrac{n_j}{n_j + n_k} d_{ij} + \dfrac{n_k}{n_j + n_k} d_{ik}$.
4. Centroid : $d_{i.jk} = \dfrac{n_j}{n_j + n_k} d_{ij} + \dfrac{n_k}{n_j + n_k} d_{ik} - \dfrac{n_j n_k}{(n_j + n_k)^2} d_{jk}$.
5. Median : $d_{i.jk} = \frac{1}{2} d_{ij} + \frac{1}{2} d_{ik} - \frac{1}{4} d_{jk}$.
6. Minimum variance : $d_{i.jk} = \{(n_i + n_j)d_{ij} + (n_i + n_k)d_{ik} - n_i d_{jk}\}/(n_i + n_j + n_k)$.

For further details see Everitt [1] or Krzanowski [2].

If the clusters are numbered 1,2,...,n then for convenience if clusters j and k, $j < k$, merge then the new cluster will be referred to as cluster j. Information on the clustering history is given by the values of j, k and d_{jk} for each of the n–1 clustering steps. In order to produce a dendrogram, the ordering of the objects such that the clusters that merge are adjacent is required. This ordering is computed so that the first element is 1. The associated distances with this ordering are also computed.

4. References

[1] EVERITT, B.
Cluster Analysis.
Heinemann, 1974.

[2] KRZANOWSKI, W.J.
Principles of Multivariate Analysis.
Oxford University Press, 1990.

5. Parameters

1: METHOD – INTEGER. *Input*

On entry: indicates which clustering method is used.

If METHOD = 1, single link.
If METHOD = 2, complete link.
If METHOD = 3, group average.
If METHOD = 4, centroid.
If METHOD = 5, median.
If METHOD = 6, minimum variance.

Constraint: METHOD = 1,2,3,4,5 or 6.

2: N – INTEGER. *Input*

On entry: the number of objects, n.

Constraint: N $\geq$ 2.

3: D(N*(N–1)/2) – ***real*** array. *Input*

On entry: the strictly lower triangle of the distance matrix. D must be stored packed by rows, i.e. $\mathrm{D}((i-1)(i-2)/2+j)$, $i > j$ must contain d_{ij}.

Constraint: $\mathrm{D}(i) \geq 0.0$, for $i = 1,2,\ldots,n(n-1)/2$.

4: ILC(N–1) – INTEGER array. *Output*

On exit: $\mathrm{ILC}(l)$ contains the number, j, of the cluster merged with cluster k (see IUC), $j < k$, at step l for $l = 1,2,\ldots,n-1$.

5: IUC(N–1) – INTEGER array. *Output*

On exit: $\mathrm{IUC}(l)$ contains the number, k, of the cluster merged with cluster j, $j < k$, at step l for $l = 1,2,\ldots,n-1$.

6: CD(N–1) – ***real*** array. *Output*

On exit: $\mathrm{CD}(l)$ contains the distance d_{jk}, between clusters j and k, $j < k$, merged at step l for $l = 1,2,\ldots,n-1$.

7: IORD(N) – INTEGER array. *Output*

On exit: the objects in dendrogram order.

8: DORD(N) – ***real*** array. *Output*

On exit: the clustering distances corresponding to the order in IORD. $\mathrm{DORD}(l)$ contains the distance at which cluster $\mathrm{IORD}(l)$ and $\mathrm{IORD}(l+1)$ merge, for $l = 1,2,\ldots,n-1$. $\mathrm{DORD}(n)$ contains the maximum distance.

9: IWK(2*N) – INTEGER array. *Workspace*

10: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, METHOD $\neq$ 1,2,3,4,5 or 6,
or N < 2.

IFAIL = 2

On entry, D(i) < 0.0 for some $i = 1,2,...,n(n-1)/2$.

IFAIL = 3

A true dendrogram cannot be formed because the distances at which clusters have merged are not increasing for all steps, i.e. CD(l) < CD$(l-1)$ for some $l = 2,3,...,n-1$. This can occur for the median and centroid methods.

7. Accuracy

For METHOD $\geq$ 3 slight rounding errors may occur in the calculations of the updated distances. These would not normally significantly affect the results, however there may be an effect if distances are (almost) equal.

If at a stage, two distances d_{ij} and d_{kl}, $i < k$ or $i = k$ and $j < l$, are equal then clusters k and l will be merged rather than clusters i and j. For single link clustering this choice will only affect the order of the objects in the dendrogram. However for other methods the choice of kl rather than ij may affect the shape of the dendrogram. If either of the distances d_{ij} or d_{kl} are affected by rounding errors then their equality, and hence the dendrogram, may be affected.

8. Further Comments

The dendrogram may be formed using G03EHF. Groupings based on the clusters formed at a given distance can be computed using G03EJF.

9. Example

Data consisting of three variables on five objects are read in. Euclidean squared distances based on two variables are computed using G03EAF, the objects are clustered using G03ECF and the dendrogram computed using G03EHF. The dendrogram is then printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03ECF Example Program Text
*     Mark 17 Revised.  NAG Copyright 1995.
*
*     .. Parameters ..
      INTEGER           NIN, NOUT
      PARAMETER         (NIN=5,NOUT=6)
      INTEGER           NMAX, MMAX, LENC
      PARAMETER         (NMAX=10,MMAX=10,LENC=20)
*     .. Local Scalars ..
      real              DMIN, DSTEP, YDIST
      INTEGER           I, IFAIL, J, LDX, M, METHOD, N, NSYM
      CHARACTER         DIST, SCALE, UPDATE
*     .. Local Arrays ..
      real              CD(NMAX-1), D(NMAX*(NMAX-1)/2), DORD(NMAX),
     +                  S(MMAX), X(NMAX,MMAX)
      INTEGER           ILC(NMAX-1), IORD(NMAX), ISX(MMAX), IUC(NMAX-1),
     +                  IWK(2*NMAX)
      CHARACTER*60      C(LENC)
      CHARACTER*3       NAME(NMAX)
*     .. External Subroutines ..
      EXTERNAL          G03EAF, G03ECF, G03EHF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03ECF Example Program Results'
```

```
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         READ (NIN,*) METHOD
         READ (NIN,*) UPDATE, DIST, SCALE
         DO 20 J = 1, N
            READ (NIN,*) (X(J,I),I=1,M), NAME(J)
   20    CONTINUE
         READ (NIN,*) (ISX(I),I=1,M)
         READ (NIN,*) (S(I),I=1,M)
*
*     Compute the distance matrix
*
         IFAIL = 0
         LDX = NMAX
*
         CALL G03EAF(UPDATE,DIST,SCALE,N,M,X,LDX,ISX,S,D,IFAIL)
*
*     Perform clustering
*
         IFAIL = 0
*
         CALL G03ECF(METHOD,N,D,ILC,IUC,CD,IORD,DORD,IWK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*)  '  Distance   Clusters Joined'
         WRITE (NOUT,*)
         DO 40 I = 1, N-1
            WRITE(NOUT,99999) CD(I), NAME(ILC(I)),NAME(IUC(I))
 40         CONTINUE
*
*     Produce dendrogram
*
         IFAIL = 0
         NSYM = LENC
         DMIN = 0.0e0
         DSTEP = (CD(N-1))/DBLE(NSYM)
*
         CALL G03EHF('S',N,DORD,DMIN,DSTEP,NSYM,C,LENC,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Dendrogram'
         WRITE (NOUT,*)
         YDIST = CD(N-1)
         DO 60 I = 1, NSYM
            IF (MOD(I,3).EQ.1) THEN
               WRITE (NOUT,99999) YDIST, C(I)
            ELSE
               WRITE (NOUT,99998) C(I)
            END IF
            YDIST = YDIST - DSTEP
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) (NAME(IORD(I)),I=1,N)
      END IF
      STOP
*
99999 FORMAT (F10.3,5X,2A)
99998 FORMAT (15X,20A)
      END
```

9.2. Program Data

```
G03ECF Example Program Data
5 3
5
'I' 'S' 'U'
 1  5.0 2.0 'A  '
 2  1.0 1.0 'B  '
 3  4.0 3.0 'C  '
 4  1.0 2.0 'D  '
 5  5.0 0.0 'E  '
 0   1   1
1.0 1.0 1.0
```

9.3. Program Results

```
G03ECF Example Program Results

  Distance   Clusters Joined

    1.000     B  D
    2.000     A  C
    6.500     A  E
   14.125     A  B

Dendrogram

   14.125           -------
                    I     I
                    I     I
   12.006           I     I
                    I     I
                    I     I
    9.887           I     I
                    I     I
                    I     I
    7.769           I     I
                 ---*     I
                 I  I     I
    5.650        I  I     I
                 I  I     I
                 I  I     I
    3.531        I  I     I
                 I  I     I
              ---*  I     I
    1.412     I  I  I  ---*
              I  I  I  I  I

              A  C  E  B  D
```

G03EFF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03EFF performs K–means cluster analysis.

2. Specification

```
      SUBROUTINE G03EFF (WEIGHT, N, M, X, LDX, ISX, NVAR, K, CMEANS, LDC,
     1                   WT, INC, NIC, CSS, CSW, MAXIT, IWK, WK, IFAIL)
      INTEGER          N, M, LDX, ISX(M), NVAR, K, LDC, INC(N), NIC(K),
     1                 MAXIT, IWK(N+3*K), IFAIL
      real             X(LDX,M), CMEANS(LDC,NVAR), WT(*), CSS(K), CSW(K),
     1                 WK(N+2*K)
      CHARACTER*1      WEIGHT
```

3. Description

Given n objects with p variables measured on each object, x_{ij} for $i = 1,2,...,n$; $j = 1,2,...,p$, G03EFF allocates each object to one of K groups or clusters to minimize the within-cluster sum of squares:

$$\sum_{k=1}^{K}\sum_{i\in S_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2,$$

where S_k is the set of objects in the kth cluster and $\bar{x}_{kj}$ is the mean for the variable j over cluster k. This is often known as K–means clustering.

In addition to the data matrix, a K by p matrix giving the initial cluster centres for the K clusters is required. The objects are then initially allocated to the cluster with the nearest cluster mean. Given the initial allocation, the procedure is to iteratively search for the K–partition with locally optimal within-cluster sum of squares by moving points from one cluster to another.

Optionally, weights for each object, w_i, can be used so that the clustering is based on within-cluster weighted sums of squares:

$$\sum_{k=1}^{K}\sum_{i\in S_k}\sum_{j=1}^{p}w_i(x_{ij}-\tilde{x}_{kj})^2,$$

where $\tilde{x}_{kj}$ is the weighted mean for variable j over cluster k.

The routine is based on the algorithm of Hartigan and Wong [2].

4. References

[1] EVERITT, B.
Cluster Analysis.
Heinemann, 1974.

[2] HARTIGAN, J.A. and WONG, M.A.
Algorithm AS 136. A K-Means clustering algorithm.
Appl. Statist., 28, Algorithm AS136, pp. 100-108, 1979.

[3] KENDALL, M. and STUART, A.
The Advanced Theory of Statistics, Vol. 3, pp. 302.
Griffin, London, 1975.

[4] KRZANOWSKI, W.J.
Principles of Multivariate Analysis.
Oxford University Press, 1990.

5. Parameters

1: WEIGHT – CHARACTER*1. *Input*

On entry: indicates if weights are to be used.
If WEIGHT = 'U' (Unweighted), then no weights are used.
If WEIGHT = 'W' (Weighted), then weights are used and must be supplied in WT.

Constraint: WEIGHT = 'U' or 'W'.

2: N – INTEGER. *Input*

On entry: the number of objects, n.

Constraint: N > 1.

3: M – INTEGER. *Input*

On entry: the total number of variables in array X.

Constraint: M ≥ NVAR.

4: X(LDX,M) – ***real*** array. *Input*

On entry: X(i,j) must contain the value of the jth variable for the ith object for $i = 1,2,...,n$; $j = 1,2,...,$M.

5: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03EFF is called.

Constraint: LDX ≥ N.

6: ISX(M) – INTEGER array. *Input*

On entry: ISX(j) indicates whether or not the jth variable is to be included in the analysis. If ISX(j) > 0, then the variable contained in the jth column of X is included, for $j = 1,2,...,$M.

Constraint: ISX(j) > 0 for NVAR values of j.

7: NVAR – INTEGER. *Input*

On entry: the number of variables included in the sums of squares calculations, p.

Constraint: 1 ≤ NVAR ≤ M.

8: K – INTEGER. *Input*

On entry: the number of clusters, K.

Constraint: K ≥ 2.

9: CMEANS(LDC,NVAR) – ***real*** array. *Input/Output*

On entry: CMEANS(i,j) must contain the value of jth variable for ith initial cluster centre, for $i = 1,2,...,K$; $j = 1,2,...,p$.

On exit: CMEANS(i,j) contains the value of jth variable for ith computed cluster centre, for $i = 1,2,...,K$; $j = 1,2,...,p$.

10: LDC – INTEGER. *Input*

On entry: the first dimension of the array CMEANS as declared in the (sub)program from which G03EFF is called.

Constraint: LDC ≥ K.

11: WT(*) – ***real*** array. *Input*

Note: the dimension of the array WT must be at least N if WEIGHT = 'W', and must be at least 1 otherwise.

On entry: if WEIGHT = 'W' then the first n elements of WT must contain the weights to be used.
If WT(i) = 0.0, then the ith observations is not included in the analysis. The effective number of observations is the sum of the weights.
If WEIGHT = 'U' then WT is not referenced and the effective number of observations is n.

Constraint: if WEIGHT = 'W' then WT(i) $\geq$ 0.0, for $i = 1,2,...,n$. and WT(i) > 0.0 for at least two values of i.

12: INC(N) – INTEGER array. *Output*

On exit: INC(i) contains the cluster to which the ith object has been allocated, for $i = 1,2,...,n$.

13: NIC(K) – INTEGER array. *Output*

On exit: NIC(i) contains the number of objects in the ith cluster, for $i = 1,2,...,K$.

14: CSS(K) – ***real*** array. *Output*

On exit: CSS(i) contains the within-cluster (weighted) sum of squares of the ith cluster, for $i = 1,2,...,K$.

15: CSW(K) – ***real*** array. *Output*

On exit: CSW(i) contains the within-cluster sum of weights of the ith cluster, for $i = 1,2,...,K$. If WEIGHT = 'U' the sum of weights is the number of objects in the cluster.

16: MAXIT – INTEGER. *Input*

On entry: the maximum number of iterations allowed in the analysis.

Constraint: MAXIT > 0.

Suggested value: MAXIT = 10.

17: IWK(N+3*K) – INTEGER array. *Workspace*

18: WK(N+2*K) – ***real*** array. *Workspace*

19: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, WEIGHT ≠ 'W' or 'U',
or N < 2,
or NVAR < 1,
or M < NVAR,
or K < 2,
or LDX < N,
or LDC < K,

or MAXIT $\leq$ 0.

IFAIL = 2

On entry, WEIGHT = 'W' and a value of WT(i) < 0.0 for some i,
or WEIGHT = 'W' and WT(i) = 0.0 for all or all but one values of i.

IFAIL = 3

On entry, the number of positive values in ISX does not equal NVAR.

IFAIL = 4

On entry, at least one cluster is empty after the initial assignment. Try a different set of initial cluster centres in CMEANS and also consider decreasing the value of K. The empty clusters may be found by examining the values in NIC.

IFAIL = 5

Convergence has not been achieved within the maximum number of iterations given by MAXIT. Try increasing MAXIT and, if possible, use the returned values in CMEANS as the initial cluster centres.

7. Accuracy

The routine produces clusters that are locally optimal; the within-cluster sum of squares may not be decreased by transfering a point from one cluster to another, but different partitions may have the same or smaller within-cluster sum of squares.

8. Further Comments

The time per iteration is approximately proportional to npK.

9. Example

The data consists of observations of five variables on twenty soils (Kendall and Stuart, [2]). The data is read in, the K–means clustering performed and the results printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03EFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, KMAX
      PARAMETER        (NMAX=20,MMAX=5,KMAX=3)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, K, LDC, LDX, M, MAXIT, N, NVAR
      CHARACTER        WEIGHT
*     .. Local Arrays ..
      real             CMEANS(KMAX,MMAX), CSS(MMAX), CSW(MMAX),
     +                 WK(NMAX+2*KMAX), WT(NMAX), X(NMAX,MMAX)
      INTEGER          INC(NMAX), ISX(MMAX), IWK(NMAX+3*KMAX), NIC(MMAX)
*     .. External Subroutines ..
      EXTERNAL         G03EFF
*     .. Executable Statements ..
*
      WRITE (NOUT,*) 'G03EFF Example Program Results'
*     Skip heading in the data file
      READ (NIN,*)
      READ (NIN,*) WEIGHT, N, M, NVAR, K, MAXIT
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
```

```
            DO 20 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M), WT(I)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) (X(I,J),J=1,M)
   40       CONTINUE
         END IF
         DO 60 I = 1, K
            READ (NIN,*) (CMEANS(I,J),J=1,NVAR)
   60    CONTINUE
         READ (NIN,*) (ISX(J),J=1,M)
         LDX = NMAX
         LDC = KMAX
         IFAIL = 0
*
         CALL G03EFF(WEIGHT,N,M,X,LDX,ISX,NVAR,K,CMEANS,LDC,WT,INC,NIC,
     +               CSS,CSW,MAXIT,IWK,WK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' The cluster each point belongs to'
         WRITE (NOUT,99999) (INC(I),I=1,N)
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' The number of points in each cluster'
         WRITE (NOUT,99999) (NIC(I),I=1,K)
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +     ' The within-cluster sum of weights of each cluster'
         WRITE (NOUT,99998) (CSW(I),I=1,K)
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +     ' The within-cluster sum of squares of each cluster'
         WRITE (NOUT,99997) (CSS(I),I=1,K)
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' The final cluster centres'
         WRITE (NOUT,*)
     +     '                 1       2       3       4       5'
         DO 80 I = 1, K
            WRITE (NOUT,99996) I, (CMEANS(I,J),J=1,NVAR)
   80    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,10I6)
99998 FORMAT (1X,5F9.2)
99997 FORMAT (1X,5F13.4)
99996 FORMAT (1X,I5,5X,5F8.4)
      END
```

9.2. Program Data

```
G03EFF Example Program Data

'u' 20 5 5 3 10              : WEIGHT N M NVAR K MAXIT

77.3 13.0  9.7 1.5 6.4
82.5 10.0  7.5 1.5 6.5
66.9 20.6 12.5 2.3 7.0
47.2 33.8 19.0 2.8 5.8
65.3 20.5 14.2 1.9 6.9
83.3 10.0  6.7 2.2 7.0
81.6 12.7  5.7 2.9 6.7
47.8 36.5 15.7 2.3 7.2
48.6 37.1 14.3 2.1 7.2
61.6 25.5 12.9 1.9 7.3
58.6 26.5 14.9 2.4 6.7
69.3 22.3  8.4 4.0 7.0
61.8 30.8  7.4 2.7 6.4
67.7 25.3  7.0 4.8 7.3
57.2 31.2 11.6 2.4 6.5
```

```
67.2 22.7 10.1 3.3 6.2
59.2 31.2  9.6 2.4 6.0
80.2 13.2  6.6 2.0 5.8
82.2 11.1  6.7 2.2 7.2
69.7 20.7  9.6 3.1 5.9

82.5 10.0  7.5 1.5 6.5          : CMEANS
47.8 36.5 15.7 2.3 7.2
67.2 22.7 10.1 3.3 6.2

1 1 1 1 1                       : ISX
```

9.3. Program Results

```
 G03EFF Example Program Results

  The cluster each point belongs to
      1     1     3     2     3     1     1     2     2     3
      3     3     3     3     3     3     3     1     1     3

  The number of points in each cluster
      6     3    11

  The within-cluster sum of weights of each cluster
      6.00      3.00     11.00

  The within-cluster sum of squares of each cluster
       46.5717       20.3800      468.8964

  The final cluster centres
                    1       2       3        4       5
     1      81.1833 11.6667  7.1500  2.0500  6.6000
     2      47.8667 35.8000 16.3333  2.4000  6.7333
     3      64.0455 25.2091 10.7455  2.8364  6.6545
```

G03EHF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G03EHF produces a dendrogram from the results of G03ECF.

2 Specification

```
      SUBROUTINE G03EHF(ORIENT, N, DORD, DMIN, DSTEP, NSYM, C, LENC,
     1                  IFAIL)
      INTEGER           N, NSYM, LENC, IFAIL
      real              DORD(N), DMIN, DSTEP
      CHARACTER*1       ORIENT
      CHARACTER*(*)     C(LENC)
```

3 Description

Hierarchical cluster analysis as performed by G03ECF can be represented by a tree that shows at which distance the clusters merge. Such a tree is known as a dendrogram. See Everitt [1] and Krzanowski [2] for examples of dendrograms. A simple example is,

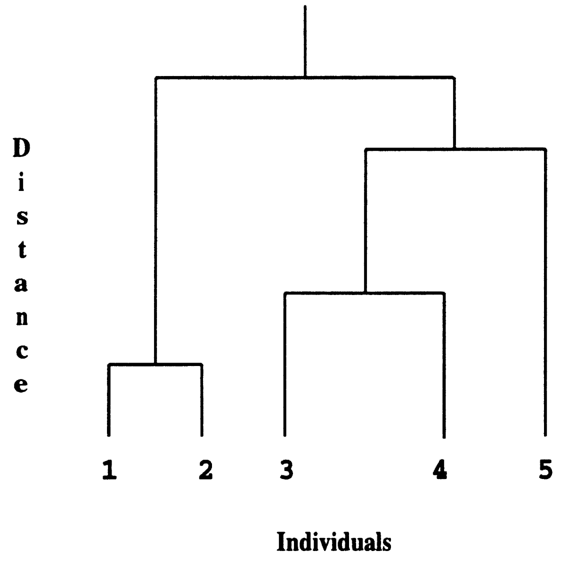


Figure 1

The end-points of the dendrogram represent the objects that have been clustered. They should be in a suitable order as given by G03ECF. Object 1 is always the first object. In the example above the height represents the distance at which the clusters merge.

The dendrogram is produced in a character array using the ordering and distances provided by G03ECF. Suitable characters are used to represent parts of the tree.

There are four possible orientations for the dendrogram. The example above has the end-points at the bottom of the diagram which will be referred to as south. If the dendrogram was the other way around with the end-points at the top of the diagram then the orientation would be north. If the end-points are at the left-hand or right-hand side of the diagram the orientation is west or east. Different symbols are used for east/west and north/south orientations.

4 References

[1] Everitt B S (1974) *Cluster Analysis* Heinemann

[2] Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

5 Parameters

1: ORIENT — CHARACTER*1 *Input*

On entry: indicates which orientation the dendrogram is to take.

If ORIENT = 'N' then the end-points of the dendrogram are to the north.

If ORIENT = 'S' then the end-points of the dendrogram are to the south.

If ORIENT = 'E' then the end-points of the dendrogram are to the east.

If ORIENT = 'W' then the end-points of the dendrogram are to the west.

Constraint: ORIENT = 'N', 'S', 'E' or 'W'.

2: N — INTEGER *Input*

On entry: the number of objects in the cluster analysis.

Constraint: N $\geq$ 2.

3: DORD(N) — ***real*** array *Input*

On entry: the array DORD as output by G03ECF. DORD contains the distances, in dendrogram order, at which clustering takes place.

Constraint: DORD(N) $\geq$ DORD(i) for $i = 1, 2, \ldots$,N$-$1.

4: DMIN — ***real*** *Input*

On entry: the clustering distance at which the dendrogram begins.

Constraint: DMIN $\geq$ 0.0.

5: DSTEP — ***real*** *Input*

On entry: the distance represented by one symbol of the dendrogram.

Constraint: DSTEP > 0.0.

6: NSYM — INTEGER *Input*

On entry: the number of character positions used in the dendrogram. Hence the clustering distance at which the dendrogram terminates is given by DMIN + NSYM $\times$ DSTEP.

Constraint: NSYM $\geq$ 1.

7: C(LENC) — CHARACTER*(*) *Output*

Note. The length of each element of C must be at least 3 $\times$ N if ORIENT = 'N' or 'S', or at least NSYM if ORIENT = 'E' or 'W'.

On exit: the elements of C contain consecutive lines of the dendrogram.

8: LENC — INTEGER *Input*

On entry: the dimension of the array C as declared in the (sub)program from which is called.

Constraints:

If ORIENT = 'N' or 'S', LENC $\geq$ NSYM,
if ORIENT = 'E' or 'W', LENC $\geq$ N.

9: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, $N < 2$,
or NSYM < 1,
or DMIN < 0.0,
or DSTEP ≤ 0.0,
or ORIENT ≠ 'N', 'S', 'E', or 'W',
or ORIENT = 'N' or 'S', LENC < NSYM,
or ORIENT = 'E' or 'W', LENC < N,
or the number of characters that can be stored in each element of array C is insufficient for the requested orientation.

IFAIL = 2

On entry, DORD(N) < DORD(i), for some $i = 1,2,\ldots$,N−1.

7 Accuracy

Not applicable.

8 Further Comments

The scale of the dendrogram is controlled by DSTEP. The smaller the value DSTEP is, the greater the amount of detail that will be given but NSYM will have to be larger to give the full dendrogram. The range of distances represented by the dendrogram is DMIN to NSYM × DSTEP. The values of DMIN, DSTEP and NSYM can thus be set so that only part of the dendrogram is produced.

The dendrogram does not include any labelling of the objects. The user can print suitable labels using the ordering given by the array IORD returned by G03ECF.

9 Example

Data consisting of three variables on five objects are read in. Euclidean squared distances are computed using G03EAF and median clustering performed by G03ECF. G03EHF is used to produce a dendrogram with orientation east and a dendrogram with orientation south. The two dendrograms are printed.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03EHF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, LDC
      PARAMETER        (NMAX=10,MMAX=10,LDC=100)
*     .. Local Scalars ..
      real             DMIN, DSTEP
      INTEGER          I, IFAIL, J, LDX, M, METHOD, N, NSYM
      CHARACTER        DIST, SCALE, UPDATE
*     .. Local Arrays ..
      real             CD(NMAX-1), D(NMAX*(NMAX-1)/2), DORD(NMAX),
     +                 S(MMAX), X(NMAX,MMAX)
      INTEGER          ILC(NMAX-1), IORD(NMAX), ISX(MMAX), IUC(NMAX-1),
     +                 IWK(2*NMAX)
      CHARACTER*50     C(LDC)
*     .. External Subroutines ..
      EXTERNAL         G03EAF, G03ECF, G03EHF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03EHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         READ (NIN,*) METHOD
         READ (NIN,*) UPDATE, DIST, SCALE
         DO 20 J = 1, N
            READ (NIN,*) (X(J,I),I=1,M)
   20    CONTINUE
         READ (NIN,*) (ISX(I),I=1,M)
         READ (NIN,*) (S(I),I=1,M)
         READ (NIN,*) DMIN, DSTEP, NSYM
*
*        Compute the distance matrix
*
         IFAIL = 0
         LDX = NMAX
*
         CALL G03EAF(UPDATE,DIST,SCALE,N,M,X,LDX,ISX,S,D,IFAIL)
*
*        Perform clustering
*
         IFAIL = 0
*
         CALL G03ECF(METHOD,N,D,ILC,IUC,CD,IORD,DORD,IWK,IFAIL)
*
*        Produce dendrograms
*
         IFAIL = 0
*
         CALL G03EHF('E',N,DORD,DMIN,DSTEP,NSYM,C,LDC,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Dendrogram, Orientation East'
         DO 40 I = 1, N
            WRITE (NOUT,*) C(I)
   40    CONTINUE
*
         READ (NIN,*) DMIN, DSTEP, NSYM
         IFAIL = 0
*
         CALL G03EHF('S',N,DORD,DMIN,DSTEP,NSYM,C,LDC,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Dendrogram, Orientation South'
         DO 60 I = 1, NSYM
            WRITE (NOUT,*) C(I)
   60    CONTINUE
      END IF
      STOP
*
      END
```

9.2 Program Data

```
G03EHF Example Program Data
5 3
5
'I' 'S' 'U'
1 1.0 1.0
2 1.0 2.0
3 6.0 3.0
4 8.0 2.0
5 8.0 0.0
0 1 1
1 1 1
0.0 1.1 40
0.0 1.0 40
```

9.3 Program Results

```
 G03EHF Example Program Results

 Dendrogram, Orientation East

          ...............................(
         (                         .......
         (                        (    ...
         (........................(...(...
Dendrogram, Orientation South



   ----------
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I        I
   I  ------*
   I  I     I
   I  I     I
   I  I     I
   I  I  ---*
   I  I  I  I
   I  I  I  I
---*  I  I  I
I  I  I  I  I
```

G03EJF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G03EJF computes a cluster indicator variable from the results of G03ECF.

2 Specification

```
SUBROUTINE G03EJF(N, CD, IORD, DORD, K, DLEVEL, IC, IFAIL)
INTEGER           N, IORD(N), K, IC(N), IFAIL
real              CD(N-1), DORD(N), DLEVEL
```

3 Description

Given a distance or dissimilarity matrix for n objects, cluster analysis aims to group the n objects into a number of more or less homogeneous groups or clusters. With agglomerative clustering methods (see G03ECF) a hierarchical tree is produced by starting with n clusters each with a single object and then at each of $n-1$ stages merging two clusters to form a larger cluster until all objects are in a single cluster. G03EJF takes the information from the tree and produces the clusters that exist at a given distance. This is equivalent to taking the dendrogram (see G03EHF) and drawing a line across at a given distance to produce clusters.

As an alternative to giving the distance at which clusters are required, the user can specify the number of clusters required and G03EJF will compute the corresponding distance. However, it may not be possible to compute the number of clusters required due to ties in the distance matrix.

If there are k clusters then the indicator variable will assign a value between 1 and k to each object to indicate to which cluster it belongs. Object 1 always belongs to cluster 1.

4 References

[1] Everitt B S (1974) *Cluster Analysis* Heinemann

[2] Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

5 Parameters

1: N — INTEGER *Input*

On entry: the number of objects, n.

Constraint: $\text{N} \geq 2$.

2: CD(N−1) — ***real*** array *Input*

On entry: the clustering distances in increasing order as returned by G03ECF.

Constraint: $\text{CD}(i+1) \geq \text{CD}(i)$, $i = 1, 2, \ldots, \text{N}-2$.

3: IORD(N) — INTEGER array *Input*

On entry: the objects in dendrogram order as returned by G03ECF.

4: DORD(N) — ***real*** array *Input*

On entry: the clustering distances corresponding to the order in IORD.

5: K — INTEGER *Input/Output*

On entry: indicates if a specified number of clusters is required.

If $K > 0$ then G03EJF will attempt to find K clusters.

If $K \le 0$ then G03EJF will find the clusters based on the distance given in DLEVEL.

Constraint: $K \le N$.

On exit: the number of clusters produced, k.

6: DLEVEL — ***real*** *Input/Output*

On entry: if $K \le 0$, then DLEVEL must contain the distance at which clusters are produced. Otherwise DLEVEL need not be set.

Constraint: if $K \le 0$ then DLEVEL > 0.0.

On exit: if $K > 0$ on entry, then DLEVEL contains the distance at which the required number of clusters are found. Otherwise DLEVEL remains unchanged.

7: IC(N) — INTEGER array *Output*

On exit: IC(i) indicates to which of k clusters the ith object belongs, for $i = 1, 2, \ldots, n$.

8: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, $K > N$,
or $K \le 0$ and DLEVEL ≤ 0.0.
or $N < 2$.

IFAIL = 2

On entry, CD is not in increasing order,
or DORD is incompatible with CD.

IFAIL = 3

On entry, $K = 1$,
or $K = N$,
or DLEVEL $\ge$ CD(N − 1),
or DLEVEL $<$ CD(1).

Note. On exit with this value of IFAIL the trivial clustering solution is returned.

IFAIL = 4

The precise number of clusters requested is not possible because of tied clustering distances. The actual number of clusters, less than the number requested, is returned in K.

7 Accuracy

The accuracy will depend upon the accuracy of the distances in CD and DORD (see G03ECF).

8 Further Comments

A fixed number of clusters can be found using the non-hierarchical method used in G03EFF.

9 Example

Data consisting of three variables on five objects are input. Euclidean squared distances are computed using G03EAF and median clustering performed using G03ECF. A dendrogram is produced by G03EHF and printed. G03EJF finds two clusters and the results are printed.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03EJF Example Program Text
*     Mark 18 Revised.  NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, LENC
      PARAMETER        (NMAX=10,MMAX=10,LENC=20)
*     .. Local Scalars ..
      real             DLEVEL, DMIN, DSTEP, YDIST
      INTEGER          I, IFAIL, J, K, LDX, M, METHOD, N, NSYM
      CHARACTER        DIST, SCALE, UPDATE
*     .. Local Arrays ..
      real             CD(NMAX-1), D(NMAX*(NMAX-1)/2), DORD(NMAX),
     +                 S(MMAX), X(NMAX,MMAX)
      INTEGER          IC(NMAX), ILC(NMAX-1), IORD(NMAX), ISX(MMAX),
     +                 IUC(NMAX-1), IWK(2*NMAX)
      CHARACTER*60     C(LENC)
      CHARACTER*3      NAME(NMAX)
*     .. External Subroutines ..
      EXTERNAL         G03EAF, G03ECF, G03EHF, G03EJF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03EJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M
      IF (N.LE.NMAX .AND. M.LE.MMAX) THEN
         READ (NIN,*) METHOD
         READ (NIN,*) UPDATE, DIST, SCALE
         DO 20 J = 1, N
            READ (NIN,*) (X(J,I),I=1,M), NAME(J)
   20    CONTINUE
         READ (NIN,*) (ISX(I),I=1,M)
         READ (NIN,*) (S(I),I=1,M)
         READ (NIN,*) K, DLEVEL
*
```

```
*     Compute the distance matrix
*
         IFAIL = 0
         LDX = NMAX
*
         CALL G03EAF(UPDATE,DIST,SCALE,N,M,X,LDX,ISX,S,D,IFAIL)
*
*     Perform clustering
*
         IFAIL = 0
*
         CALL G03ECF(METHOD,N,D,ILC,IUC,CD,IORD,DORD,IWK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Distance   Clusters Joined'
         WRITE (NOUT,*)
         DO 40 I = 1, N - 1
            WRITE (NOUT,99999) CD(I), NAME(ILC(I)), NAME(IUC(I))
   40    CONTINUE
*
*     Produce dendrogram
*
         IFAIL = 0
         NSYM = LENC
         DMIN = 0.0e0
         DSTEP = (CD(N-1))/real(NSYM)
*
         CALL G03EHF('S',N,DORD,DMIN,DSTEP,NSYM,C,LENC,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Dendrogram'
         WRITE (NOUT,*)
         YDIST = CD(N-1)
         DO 60 I = 1, NSYM
            IF (MOD(I,3).EQ.1) THEN
               WRITE (NOUT,99999) YDIST, C(I)
            ELSE
               WRITE (NOUT,99998) C(I)
            END IF
            YDIST = YDIST - DSTEP
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) (NAME(IORD(I)),I=1,N)
         IFAIL = 0
*
         CALL G03EJF(N,CD,IORD,DORD,K,DLEVEL,IC,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99997) ' Allocation to ', K, ' clusters'
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Object  Cluster'
         WRITE (NOUT,*)
         DO 80 I = 1, N
            WRITE (NOUT,99996) NAME(I), IC(I)
   80    CONTINUE
      END IF
      STOP
```

```
*
99999 FORMAT (F10.3,5X,2A)
99998 FORMAT (15X,20A)
99997 FORMAT (A,I2,A)
99996 FORMAT (5X,A,5X,I2)
      END
```

9.2 Program Data

```
G03EJF Example Program Data
5 3
5
'I' 'S' 'U'
 1  5.0 2.0 'A  '
 2  1.0 1.0 'B  '
 3  4.0 3.0 'C  '
 4  1.0 2.0 'D  '
 5  5.0 0.0 'E  '
 0   1   1
1.0 1.0 1.0
2 0.0
```

9.3 Program Results

```
 G03EJF Example Program Results

   Distance   Clusters Joined

     1.000     B  D
     2.000     A  C
     6.500     A  E
    14.125     A  B

 Dendrogram

    14.125           -------
                     I     I
                     I     I
    12.006           I     I
                     I     I
                     I     I
     9.887           I     I
                     I     I
                     I     I
     7.769           I     I
                  ---*     I
                  I  I     I
     5.650        I  I     I
                  I  I     I
                  I  I     I
     3.531        I  I     I
                  I  I     I
               ---*  I     I
     1.412     I  I  I  ---*
               I  I  I  I  I

               A  C  E  B  D
```

2: N — INTEGER *Input*

On entry: the number of objects in the distance matrix, n.

Constraint: N > NDIM.

3: D(N*(N−1)/2) — ***real*** array *Input*

On entry: the lower triangle of the distance matrix D stored packed by rows. That is D($(i-1)*(i-2)/2+j$) must contain d_{ij} for $i = 2,3,\ldots,n; j = 1,2,\ldots,i-1$.

Constraint: D(i) $\geq 0.0, i = 1,2,\ldots,n(n-1)/2$.

4: NDIM — INTEGER *Input*

On entry: the number of dimensions used to represent the data, k.

Constraint: NDIM ≥ 1.

5: X(LDX,NDIM) — ***real*** array *Output*

On exit: the ith row contains k coordinates for the ith point, $i = 1,2,\ldots,n$.

6: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03FAF is called.

Constraint: LDX $\geq$ N.

7: EVAL(N) — ***real*** array *Output*

On exit: If ROOTS = 'A', EVAL contains the n scaled eigenvalues of the matrix E. If ROOTS = 'L', EVAL contains the largest k scaled eigenvalues of the matrix E. In both cases the eigenvalues are divided by the sum of the eigenvalues (that is, the trace of E).

8: WK(N*(N+17)/2−1) — ***real*** array *Workspace*
9: IWK(5*N) — INTEGER array *Workspace*
10: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL=1

On entry, NDIM < 1,
or N < NDIM,
or ROOTS $\neq$ 'A' or 'L',
or LDX < N.

IFAIL=2

On entry, D(i) < 0.0 for some $i, i = 1,2,\ldots,n(n-1)/2$,
or all elements of D = 0.0.

IFAIL=3

There are less than NDIM eigenvalues greater than zero. Try a smaller number of dimensions (NDIM) or use non-metric scaling (G03FCF).

IFAIL=4

The computation of the eigenvalues or eigenvectors has failed. Seek expert help.

7 Accuracy

The routine uses F08JFF or F08JJF to compute the eigenvalues and F08JKF to compute the eigenvectors. These routines should be consulted for a discussion of the accuracy of the computations involved.

8 Further Comments

Alternative, non-metric, methods of scaling are provided by G03FCF.

The relationship between principal coordinates and principal components, see G03FCF, is discussed in Krzanowski [3] and Gower [1].

9 Example

The data, given by Krzanowski [3], are dissimilarities between water vole populations in Europe. The first two principal co-ordinates are computed by G03FAF and then plotted using G01AGF.

9.1 Example Text

Note: the listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03FAF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NNMAX
      PARAMETER        (NMAX=14,NNMAX=NMAX*(NMAX-1)/2)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, N, NDIM, NN
      CHARACTER        ROOTS
*     .. Local Arrays ..
      real             D(NNMAX), E(NMAX), WK(NNMAX+9*NMAX), X(NMAX,NMAX)
      INTEGER          IWK(5*NMAX)
*     .. External Subroutines ..
      EXTERNAL         G01AGF, G03FAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03FAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NDIM, ROOTS
      IF (N.LE.NMAX) THEN
         NN = N*(N-1)/2
         READ (NIN,*) (D(I),I=1,NN)
         IFAIL = 0
*
         CALL G03FAF(ROOTS,N,D,NDIM,X,NMAX,E,WK,IWK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Scaled Eigenvalues'
         WRITE (NOUT,*)
         IF (ROOTS.EQ.'L' .OR. ROOTS.EQ.'l') THEN
            WRITE (NOUT,99999) (E(I),I=1,NDIM)
         ELSE
            WRITE (NOUT,99999) (E(I),I=1,N)
         END IF
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Co-ordinates'
         WRITE (NOUT,*)
         DO 20 I = 1, N
            WRITE (NOUT,99999) (X(I,J),J=1,NDIM)
   20    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Plot of first two dimensions'
         WRITE (NOUT,*)
         IFAIL = 0
         CALL G01AGF(X,X(1,2),N,IWK,50,18,IFAIL)
      END IF
      STOP
*
99999 FORMAT (8F10.4)
      END
```

9.2 Example Data

```
G03FAF Example Program Data

14 2 '1'

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 0.085 0.051
0.462 0.266 0.322 0.435 0.268 0.025
0.628 0.442 0.444 0.406 0.240 0.129 0.014
0.113 0.070 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 0.071
0.434 0.419 0.339 0.505 0.469 0.390 0.315 0.349 0.151 0.430
0.762 0.633 0.781 0.700 0.758 0.625 0.469 0.618 0.440 0.538 0.607
0.530 0.389 0.482 0.579 0.597 0.498 0.374 0.562 0.247 0.383 0.387 0.084
0.586 0.435 0.550 0.530 0.552 0.509 0.369 0.471 0.234 0.346 0.456 0.090 0.038
```

9.3 Example Results

```
 G03FAF Example Program Results

  Scaled Eigenvalues

     0.7871    0.2808

  Co-ordinates

     0.2408    0.2337
     0.1137    0.1168
     0.2394    0.0760
     0.2129    0.0605
     0.2495   -0.0693
     0.1487   -0.0778
    -0.0514   -0.1623
     0.0115   -0.3446
    -0.0039    0.0059
     0.0386   -0.0089
    -0.0421   -0.0566
 -0.5158    0.0291
 -0.3180    0.1501
 -0.3238    0.0475

Plot of first two dimensions

          +....+....+....+....+....+....+....+....+....+....+.
          .                             .                    .
          .                             .                    .
          .                             .           1        .
     0.200+                             +                    +
          .             1               .                    .
          .                             .     1              .
          .                             .          11        .
          .   1         1               .                    .
     0.000+....+....+....+....+....+....1.1..+....+....+....++
          .                           1 .                    .
          .                             .      1    1        .
          .                             .                    .
          .                          1  .                    .
    -0.200+                             +                    +
          .                             .                    .
          .                             .                    .
          .                             .                    .
          .                             .1                   .
    -0.400+....+....+....+....+....+....+....+....+....+....+.
         -.6000    -.4000    -.2000    0.0000    0.2000    0.4000
              -.5000    -.3000    -.1000    0.1000    0.3000
```

G03FCF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G03FCF performs non-metric (ordinal) multidimensional scaling.

2 Specification

```
 SUBROUTINE G03FCF(TYPE, N, NDIM, D, X, LDX, STRESS, DFIT, ITER,
1                  IOPT, WK, IWK, IFAIL)
 INTEGER           N, NDIM, LDX, ITER, IOPT,
1                  IWK(N*(N-1)/2+N*NDIM+5), IFAIL
 real              D(N*(N-1)/2), X(LDX,NDIM), STRESS,
1                  DFIT(2*N*(N-1)), WK(15*N*NDIM)
 CHARACTER*1       TYPE
```

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' are the objects i and j. If p variables x_k have been recorded for each observation this measure may be based on Euclidean distance, $d_{ij} = \sum_{k=1}^{p}(x_{ki} - x_{kj})^2$, 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 G03FAF, 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,

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

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,

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

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 G03FAF. Given an initial configuration G03FCF uses the optimization routine E04DGF to find the configuration of points that minimizes STRESS or SSTRESS. The routine E04DGF uses a conjugate gradient algorithm. G03FCF 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 routines from the G05 Chapter.

4 References

[1] Chatfield C and Collins A J (1980) *Introduction to Multivariate Analysis* Chapman and Hall

[2] Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

5 Parameters

1: TYPE — CHARACTER*1 *Input*

On entry: indicates whether STRESS or SSTRESS is to be used as the criterion.

If TYPE = 'T' STRESS is used.

If TYPE = 'S' SSTRESS is used.

Constraint: TYPE = 'S' or 'T'.

2: N — INTEGER *Input*

On entry: the number of objects in the distance matrix, n.

Constraint: N > NDIM.

3: NDIM — INTEGER *Input*

On entry: the number of dimensions used to represent the data, m.

Constraint: NDIM $\geq$ 1.

4: D(N*(N−1)/2) — ***real*** array *Input*

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

5: X(LDX,NDIM) — ***real*** array *Input/Output*

On entry: the ith row must contain an initial estimate of the co-ordinates for the ith point, $i = 1,2,\ldots,n$. One method of computing these is to use G03FAF.

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

6: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03FCF is called.

Constraint: LDX $\geq$ N.

7: STRESS — ***real*** *Output*

On exit: the value of STRESS or SSTRESS at the final iteration.

8: DFIT(2*N*(N−1)) — ***real*** array *Output*

On exit: auxiliary outputs. If TYPE = '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 the distances $\hat{d}_{ij}$ ordered by the input distances, d_{ij}, the third set of $n(n-1)/2$ elements contains the 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 TYPE = '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.

9: ITER — INTEGER *Input*

On entry: the maximum number of iterations in the optimization process. If ITER = 0 a default value of 50 is used, if ITER < 0 a default value of $\max(50, 5nm)$ (the default for E04DGF) is used.

10: IOPT — INTEGER *Input*

On entry: selects the options, other than the number of iterations, that control the optimization. If 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. If IOPT > 0, the default values are used except that the accuracy is given by 10^{-i} where $i =$ IOPT. Finally, if IOPT < 0 the option setting mechanism of E04DGF can be used to set all options except Iteration Limit; this option is only recommended for experienced users of NAG optimization routines. For further details see E04DGF.

11: WK(15*N*NDIM) — ***real*** array *Workspace*

12: IWK(N*(N−1)/2+N*NDIM+5) — INTEGER array *Workspace*

13: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL=1

On entry, NDIM < 1,
or N ≤ NDIM,
or TYPE ≠ '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 or 9 in E04DGF.

7 Accuracy

After a successful optimization the relative accuracy of STRESS should be approximately ϵ, as specified by IOPT.

8 Further Comments

The optimization routine E04DGF used by G03FCF 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 repectively. 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 E04DGF with the defaults as given by that routine.

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 routine G03FAF 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 G03FAF to give an initial set of co-ordinates.

9 Example

The data, given by Krzanowski [2], are dissimilarities between water vole populations in Europe. Initial estimates are provided by the first two principal co-ordinates computed by G03FAF. The two dimension solution is computed using G03FCF and then plotted using G01AGF.

9.1 Example Text

Note: the listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03FCF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, NNMAX
      PARAMETER        (NMAX=14,MMAX=2,NNMAX=NMAX*(NMAX-1)/2)
*     .. Local Scalars ..
      real             STRESS
      INTEGER          I, IFAIL, IOPT, ITER, J, LDX, N, NDIM, NN
      CHARACTER        TYPE
*     .. Local Arrays ..
      real             D(NNMAX), DFIT(4*NNMAX), WK(NNMAX+15*NMAX*MMAX),
     +                 X(NMAX,NMAX)
      INTEGER          IWK(NNMAX+NMAX*NMAX+5)
*     .. External Subroutines ..
      EXTERNAL         G01AGF, G03FAF, G03FCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03FCF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NDIM, TYPE
      IF (N.LE.NMAX) THEN
         NN = N*(N-1)/2
         READ (NIN,*) (D(I),I=1,NN)
         LDX = NMAX
         IFAIL = 0
         CALL G03FAF('L',N,D,NDIM,X,LDX,WK,WK(N+1),IWK,IFAIL)
         ITER = 0
         IOPT = 0
         IFAIL = 0
```

```
*
         CALL G03FCF(TYPE,N,NDIM,D,X,LDX,STRESS,DFIT,ITER,IOPT,WK,IWK,
     +               IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) STRESS
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Co-ordinates'
         WRITE (NOUT,*)
         DO 20 I = 1, N
            WRITE (NOUT,99998) (X(I,J),J=1,NDIM)
   20    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Plot of first two dimensions'
         WRITE (NOUT,*)
         IFAIL = 0
         CALL G01AGF(X(1,1),X(1,2),N,IWK,50,18,IFAIL)
      END IF
      STOP
*
99999 FORMAT (10X,'STRESS = ',e13.4)
99998 FORMAT (8F10.4)
      END
```

9.2 Example Data

```
G03FCF Example Program Data

14 2 'T'

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 0.085 0.051
0.462 0.266 0.322 0.435 0.268 0.025
0.628 0.442 0.444 0.406 0.240 0.129 0.014
0.113 0.070 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 0.071
0.434 0.419 0.339 0.505 0.469 0.390 0.315 0.349 0.151 0.430
0.762 0.633 0.781 0.700 0.758 0.625 0.469 0.618 0.440 0.538 0.607
0.530 0.389 0.482 0.579 0.597 0.498 0.374 0.562 0.247 0.383 0.387 0.084
0.586 0.435 0.550 0.530 0.552 0.509 0.369 0.471 0.234 0.346 0.456 0.090 0.038
```

9.3 Example Results

```
 G03FCF Example Program Results

          STRESS =    0.1256E+00

  Co-ordinates

    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

Plot of first two dimensions

          +....+....+....+....+....+....+....+....+....+....+.
          .                             .                    .
          .                             .                    .
          .                             .         1          .
     0.200+                             +                    +
          .              1              .    1               .
          .    1                        .                    .
          .            1                .          1         .
          .                             . 1            1     .
     0.000+....+....+....+....+....+....+....+....+....+....++
          .                             .1      1            .
          .                             .            1       .
          .                             .                    .
          .                             1                    .
    -0.200+                             +                    +
          .                     1       .                    .
          .                             .                    .
          .                             .   1                .
          .                             .                    .
    -0.400+....+....+....+....+....+....+....+....+....+....+.
         -.6000    -.4000    -.2000    0.0000    0.2000    0.4000
              -.5000    -.3000    -.1000    0.1000    0.3000
```

G03ZAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G03ZAF produces standardized values (z-scores) for a data matrix.

2. Specification

```
      SUBROUTINE G03ZAF (N, M, X, LDX, NVAR, ISX, S, E, Z, LDZ,
     1                   IFAIL)
      INTEGER            N, M, LDX, NVAR, ISX(M), LDZ, IFAIL
      real               X(LDX,M), S(M), E(M), Z(LDZ,NVAR)
```

3. Description

For a data matrix, X, consisting of n observations on p variables, with elements x_{ij}, G03ZAF computes a matrix, Z, with elements z_{ij} such that:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \qquad i = 1,2,...,n; j = 1,2,...,p,$$

where μ_j is a location shift and σ_j is a scaling factor. Typically μ_j will be the mean and σ_j will be the standard deviation of the jth variable and therefore the elements in column j of Z will have zero mean and unit variance.

4. References

None.

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of observations in the data matrix, n.

Constraint: N $\geq$ 1.

2: M – INTEGER. *Input*

On entry: the number of variables in the data array X.

Constraint: M $\geq$ NVAR.

3: X(LDX,M) – ***real*** array. *Input*

On entry: X(i,j) must contain the ith sample point for the jth variable, x_{ij}, for i = 1,2,...,n; j = 1,2,...,M.

4: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03ZAF is called.

Constraint: LDX $\geq$ N.

5: NVAR – INTEGER. *Input*

On entry: the number of variables to be standardized, p.

Constraint: NVAR $\geq$ 1.

6: ISX(M) – INTEGER array. *Input*

On entry: ISX(j) indicates whether or not the observations on the jth variable are included in the matrix of standardized values.

If ISX(j) $\neq$ 0, then the observations from the jth variable are included.

If ISX(j) = 0, then the observations from the jth variable are not included.

Constraint: ISX(j) $\neq$ 0 for NVAR values of j.

7: S(M) – ***real*** array. *Input*

On entry: if ISX(j) $\neq$ 0, then S(j) must contain the scaling (standard deviation), σ_j, for the jth variable.

If ISX(j) = 0, then S(j) is not referenced.

Constraint: if ISX(j) $\neq$ 0, then S(j) > 0.0 for j = 1,2,...,M.

8: E(M) – ***real*** array. *Input*

On entry: if ISX(j) $\neq$ 0, then E(j) must contain the location shift (mean), μ_j, for the jth variable.

If ISX(j) = 0, then E(j) is not referenced.

9: Z(LDZ,NVAR) – ***real*** array. *Output*

On exit: the matrix of standardized values (z-scores), Z.

10: LDZ – INTEGER. *Input*

On entry: the first dimension of the array Z as declared in the (sub)program from which G03ZAF is called.

Constraint: LDZ $\geq$ N.

11: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, N < 1,
or NVAR < 1,
or M < NVAR,
or LDX < N,
or LDZ < N.

IFAIL = 2

On entry, there are not precisely NVAR elements of ISX $\neq$ 0.

IFAIL = 3

On entry, ISX(j) $\neq$ 0 and S(j) $\leq$ 0.0 for some j.

7. Accuracy

Standard accuracy is achieved.

8. Further Comments

Means and standard deviations may be obtained using G01AAF or G02BXF.

9. Example

A 4 by 3 data matrix is input along with location and scaling values. The first and third columns are scaled and the results printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G03ZAF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX
      PARAMETER        (NMAX=4,MMAX=3)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, M, N, NVAR
*     .. Local Arrays ..
      real             E(MMAX), S(MMAX), X(NMAX,MMAX), Z(NMAX,MMAX)
      INTEGER          ISX(MMAX)
*     .. External Subroutines ..
      EXTERNAL         G03ZAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G03ZAF Example Program Results'
*     Skip headings in data file
      READ (NIN,*)
      READ (NIN,*) N, M, NVAR
      IF (M.LE.MMAX .AND. N.LE.NMAX) THEN
         DO 20 I = 1, N
            READ (NIN,*) (X(I,J),J=1,M)
   20    CONTINUE
         READ (NIN,*) (ISX(J),J=1,M)
         READ (NIN,*) (E(J),J=1,M)
         READ (NIN,*) (S(J),J=1,M)
         IFAIL = 0
*
         CALL G03ZAF(N,M,X,NMAX,NVAR,ISX,S,E,Z,NMAX,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Standardized Values'
         DO 40 I = 1, N
            WRITE (NOUT,99999) (Z(I,J),J=1,NVAR)
   40    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,9F8.3)
      END
```

9.2. Program Data

```
G03ZAF Example Program Data
4 3 2
15.0 0.0 1500.0
12.0 1.0 1000.0
18.0 2.0 1200.0
14.0 3.0  500.0
 1    0    1
14.75 0.0 1050.0
 2.50 0.0  420.3
```

9.3. Program Results

```
G03ZAF Example Program Results

 Standardized Values
   0.100   1.071
  -1.100  -0.119
   1.300   0.357
  -0.300  -1.309
```

Chapter G04 – Analysis of Variance

Note. Please refer to the Users' Note for your implementation to check that a routine is available.

Routine Name	Mark of Introduction	Purpose
G04AGF	8	Two-way analysis of variance, hierarchical classification, subgroups of unequal size
G04BBF	16	Analysis of variance, randomized block or completely randomized design, treatment means and standard errors
G04BCF	17	Analysis of variance, general row and column design, treatment means and standard errors
G04CAF	16	Analysis of variance, complete factorial design, treatment means and standard errors
G04DAF	17	Computes sum of squares for contrast between means
G04DBF	17	Computes confidence intervals for differences between means computed by G04BBF or G04BCF
G04EAF	17	Computes orthogonal polynomials or dummy variables for factor/classification variable

Chapter G04

Analysis of Variance

Contents

1 Scope of the Chapter

This chapter is concerned with methods for analysing the results of designed experiments. The range of experiments covered includes:

(1) single factor designs with equal sized blocks such as randomised complete block and balanced incomplete block designs,
(2) row and column designs such as Latin squares, and
(3) complete factorial designs.

Further designs may be analysed by combining the analyses provided by multiple calls to routines or by using general linear model routines provided in Chapter G02.

2 Background to the Problems

2.1 Experimental Designs

An experimental design consists of a plan for allocating a set of controlled conditions, the treatments, to subsets of the experimental material, the plots or units. Two examples are:

(a) In an experiment to examine the effects of different diets on the growth of chickens, the chickens were kept in pens and a different diet was fed to the birds in each pen. In this example the pens are the units and the different diets are the treatments.

(b) In an experiment to compare four materials for ware-loss, a sample from each of the materials is tested in a machine that simulates ware. The machine can take four samples at a time and a number of runs are made. In this experiment the treatments are the materials and the units are the samples from the materials.

In designing an experiment the following principles are important.

(1) Randomisation: Given the overall plan of the experiment, the final allocation of treatments to units is performed using a suitable random allocation. This avoids the possibility of a systematic bias in the allocation and gives a basis for the statistical analysis of the experiment.

(2) Replication: Each treatment should be 'observed' more than once. So in example (b) more than one sample from each material should be tested. Replication allows for an estimate of the variablity of the treatment effect to be measured.

(3) Blocking: In many situations the experimental material will not be homogeneous and there may be some form of systematic variation in the experimental material. In order to reduce the effect of systematic variation the material can be grouped into blocks so that units within a block are similar but there is variation between blocks. For example, in an animal experiment litters may be considered as blocks; in an industrial experiment it may be material from one production batch.

(4) Factorial designs: If more than one type of treatment is under consideration, for example the effect of changes in temperature and changes in pressure, a factorial design consists of looking at all combinations of temperature and pressure. The different types of treatment are known as factors and the different values of the factors that are considered in the experiment are known as levels. So if three temperatures and four different pressures were being considered, then factor 1 (temperature) would have 3 levels and factor 2 (pressure) would have four levels and the design would be a 3×4 factorial giving a total of 12 treatment combinations. This design has the advantage of being able to detect the interaction between factors, that is, the effect of the combination of factors.

The following are examples of standard experimental designs; in the descriptions, it is assumed that there are t treatments.

(1) Completely Randomised Design: There are no blocks and the treatments are allocated to units at random.

(2) Randomised Complete Block Design: The experimental units are grouped into b blocks of t units and each treatment occurs once in each block. The treatments are allocated to units within blocks at random.

(3) Latin Square Designs: The units can be represented as cells of a $t \times t$ square classified by rows and columns. The t rows and t columns represent sources of variation in the experimental material. The design allocates the treatments to the units so that each treatment occurs once in each row and each column.

(4) Balanced Incomplete Block Designs: The experimental units are grouped into b blocks of $k < t$ units. The treatments are allocated so that each treatment is replicated the same number of times and each treatment occurs in the same block with any other treatment the same number of times. The treatments are allocated to units within blocks at random.

(5) Complete Factorial Experiments: If there are t treatment combinations derived from the levels of all factors then either there are no blocks or the blocks are of size t units.

Other designs include: partially balanced incomplete block designs, split-plot designs, factorial designs with confounding, and fractional factorial designs. For further information on these designs, see Cochran and Cox [1], Davies [2] or John and Quenouille [4].

2.2 Analysis of Variance

The analysis of a designed experiment usually consists of two stages. The first is the computation of the estimate of variance of the underlying random variation in the experiment along with tests for the overall effect of treatments. This results in an analysis of variance (ANOVA) table. The second stage is a more detailed examination of the effect of different treatments either by comparing the difference in treatment means with an appropriate standard error or by the use of orthogonal contrasts.

The analysis assumes a linear model such as:

$$y_{ij} = \mu + \delta_i + \tau_l + e_{ij}$$

where y_{ij} is the observed value for unit j of block i, μ is the overall mean, δ_i is the effect of the ith block, τ_l is the effect of the lth treatment which has been applied to the unit, and e_{ij} is the random error term associated with this unit. The expected value of e_{ij} is zero and its variance is σ^2.

In the analysis of variance, the total variation, measured by the sum of squares of observations about the overall mean, is partitioned into the sum of squares due to blocks, the sum of squares due to treatments, and a residual or error sum of squares. This partition corresponds to the parameters β, τ and σ. In parallel to the partition of the sum of squares there is a partition of the degrees of freedom associated with the sums of squares. The total degrees of freedom is $n - 1$, where n is the number of observations. This is partitioned into $b - 1$ degrees of freedom for blocks, $t - 1$ degrees of freedom for treatments, and $n - t - b + 1$ degrees of freedom for the residual sum of squares. From these the mean squares can be computed as the sums of squares divided by their degrees of freedom. The residual mean square is an estimate of σ^2. An F-test for an overall effect of the treatments can be calculated as the ratio of the treatment mean square to the residual mean square.

For row and column designs the model is:

$$y_{ij} = \mu + \rho_i + \gamma_j + \tau_l + e_{ij}$$

where ρ_i is the effect of the ith row and γ_j is the effect of the jth column. Usually the rows and columns are orthogonal. In the analysis of variance the total variation is partitioned into rows, columns treatments and residual.

In the case of factorial experiments, the treatment sum of squares and degrees of freedom may be partitioned into main effects for the factors and interactions between factors. The main effect of a factor is the effect of the factor averaged over all other factors. The interaction between two factors is the additional effect of the combination of the two factors, over and above the additive effects of the two factors, averaged over all other factors. For a factorial experiment in blocks with two factors, A and B, in which the jth unit of the ith block received level l of factor A and level k of factor B the model is:

$$y_{ij} = \mu + \delta_i + (\alpha_l + \beta_k + \alpha\beta_{lk}) + e_{ij}$$

where α_l is the main effect of level l of factor a, β_k is the main effect of level k of factor B, and $\alpha\beta_{lk}$ is the interaction between level l of A and level k of B. Higher-order interactions can be defined in a similar way.

Once the signifcant treatment effects have been uncovered they can be further investigated by comparing the differences between the means with the appropriate standard error. Some of the assumptions of the analysis can be checked by examining the residuals.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

The Chapter contains routines that can handle a wide range of experimental designs plus routines for further analysis and a routine to compute dummy variables for use in a general linear model.

G04BBF	computes the analysis of variance and treatment means with standard errors for any block design with equal sized blocks. The routine will handle both complete block designs and balanced and partially balanced incomplete block designs.
G04BCF	computes the analysis of variance and treatment means with standard errors for a row and column designs such as a Latin square.
G04CAF	computes the analysis of variance and treatment means with standard errors for a complete factorial experiment.

Other designs can be analysed by combinations of calls to G04BBF, G04BCF and G04CAF. The routines compute the residuals from the model specified by the design, so these can then be input as the response variable in a second call to one of the routines. For example a factorial experiment in a Latin square design can be analysed by first calling G04BCF to remove the row and column effects and then calling G04CAF with the residuals from G04BCF as the response variable to compute the ANOVA for the treatments. Another example would be to use both G02DAF and G04BBF to compute an analysis of covariance.

It is also possible to analyse factorial experiments in which some effects have been confounded with blocks or some fractional factorial experiments. For examples see Morgan [6].

For experiments with missing values, these values can be estimated by using the Healy and Westmacott procedure, see John and Quenouille [4]. This procedure involves starting with initial estimates for the missing values and then making adjustments based on the residuals from the analysis. The improved estimates are then used in further iterations of the process.

For designs that cannot be analysed by the above approach the routine G04EAF can be used to compute dummy variables from the classification variables or factors that define the design. These dummy variables can then be used with the general linear model routine G02DAF.

As well as the routines considered above the routine G04AGF computes the analysis of variance for a two strata nested design.

In addition to the routines for computing the means and the basic analysis of variance two routines are available for further analysis.

G04DAF	computes the sum of squares for a user defined contrast between means. For example, if there are four treatments, the first is a control and the other three are different amounts of a chemical the contrasts that are the difference between no chemical and chemical and the linear effect of chemical could be defined. G04DAF could be used to compute the sums of squares for these contrasts from which the appropriate F-tests could be computed.
G04DBF	computes simultaneous confidence intervals for the differences between means with the choice of different methods such as the Tukey–Kramer, Bonferron and Dunn–Sidak.

4 Routines Withdrawn or Scheduled for Withdrawal

Since Mark 13 the following routines have been withdrawn. Advice on replacing calls to these routines is given in the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines'.

G04ADF	G04AEF	G04AFF

5 References

[1] Cochran W G and Cox G M (1957) *Experimental Designs* Wiley

[2] Davis O L (1978) *The Design and Analysis of Industrial Experiments* Longman

[3] John J A (1987) *Cyclic Designs* Chapman and Hall

[4] John J A and Quenouille M H (1977) *Experiments: Design and Analysis* Griffin

[5] Searle S R (1971) *Linear Models* Wiley

[6] Morgan G W (1993) Analysis of variance using the NAG Fortran Library: Examples from Cochran and Cox *NAG Technical Report TR 3/93* NAG Ltd, Oxford

G04AGF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G04AGF performs an analysis of variance for a two-way hierarchical classification with subgroups of possibly unequal size, and also computes the treatment group and subgroup means. A fixed effects model is assumed.

2. Specification

```
      SUBROUTINE G04AGF (Y, N, K, LSUB, NOBS, L, NGP, GBAR, SGBAR, GM, SS,
     1                   IDF, F, FP, IFAIL)
      INTEGER            N, K, LSUB(K), NOBS(L), L, NGP(K), IDF(4), IFAIL
      real               Y(N), GBAR(K), SGBAR(L), GM, SS(4), F(2), FP(2)
```

3. Description

In a two-way hierarchical classification, there are k (≥ 2) treatment groups, the ith of which is subdivided into l_i treatment subgroups. The jth subgroup of group i contains n_{ij} observations, which may be denoted by

$$y_{1ij}, y_{2ij}, \ldots, y_{n_{ij}ij}.$$

The general observation is denoted by y_{mij}, being the mth observation in subgroup j of group i, for $1 \leq i \leq k$, $1 \leq j \leq l_i$, $1 \leq m \leq n_{ij}$.

The following quantities are computed:

(i) The subgroup means

$$\bar{y}_{.ij} = \frac{\sum_{m=1}^{n_{ij}} y_{mij}}{n_{ij}}$$

(ii) The group means

$$\bar{y}_{.i.} = \frac{\sum_{j=1}^{l_i} \sum_{m=1}^{n_{ij}} y_{mij}}{\sum_{j=1}^{l_i} n_{ij}}$$

(iii) The grand mean

$$\bar{y}_{...} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l_i} \sum_{m=1}^{n_{ij}} y_{mij}}{\sum_{i=1}^{k} \sum_{j=1}^{l_i} n_{ij}}$$

(iv) The number of observations in each group

$$n_{i.} = \sum_{j=1}^{l_i} n_{ij}$$

(v) Sums of squares.

Between groups $= SS_g = \sum_{i=1}^{k} n_{i.} (\bar{y}_{.i.} - \bar{y}_{...})^2$

Between subgroups within groups $= SS_{sg} = \sum_{i=1}^{k} \sum_{j=1}^{l_i} n_{ij} (\bar{y}_{.ij} - \bar{y}_{.i.})^2$

$$\text{Residual (within subgroups)} = SS_{res} = \sum_{i=1}^{k}\sum_{j=1}^{l_i}\sum_{m=1}^{n_{ij}} (y_{mij}-\bar{y}_{.ij})^2$$
$$= SS_{tot} - SS_g - SS_{sg}$$

$$\text{Corrected total} = SS_{tot} = \sum_{i=1}^{k}\sum_{j=1}^{l_i}\sum_{m=1}^{n_{ij}} (y_{mij}-\bar{y}_{...})^2$$

(vi) Degrees of freedom of variance components.

Between groups	: $k - 1$
Subgroups within groups	: $l - k$
Residual	: $n - l$
Total	: $n - 1$

where $l = \sum_{i=1}^{k} l_i$,

$n = \sum_{i=1}^{k} n_{i.}$

(vii) F ratios. These are the ratios of the group and subgroup mean squares to the residual mean square.

Groups

$$F_1 = \frac{\text{Between groups sum of squares}/(k-1)}{\text{Residual sum of squares}/(n-l)} = \frac{SS_g/(k-1)}{SS_{res}/(n-l)}$$

Subgroups

$$F_2 = \frac{\text{Between subgroups (within group) sum of squares}/(l-k)}{\text{Residual sum of squares}/(n-l)} = \frac{SS_{sg}/(l-k)}{SS_{res}/(n-l)}$$

If either F ratio exceeds 9999.0, the value 9999.0 is assigned instead.

(viii) F significances. The probability of obtaining a value from the appropriate F-distribution which exceeds the computed mean square ratio.

Groups

$$p_1 = \text{Prob}(F_{(k-1),(n-l)} > F_1)$$

Subgroups

$$p_2 = \text{Prob}(F_{(l-k),(n-l)} > F_2)$$

where F_{v_1,v_2} denotes the central F-distribution with degrees of freedom v_1 and v_2 .

If any $F_i = 9999.0$, then p_i is set to zero, $i = 1,2$.

4. References

[1] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics.
Griffin, London, 3, pp. 34-35, 1976.

[2] MOORE, P.G., SHIRLEY, E.A. and EDWARD, D.E.
Standard Statistical Calculations.
Pitman, p. 31, 1972.

5. Parameters

1: Y(N) – ***real*** array. *Input*

On entry: the elements of Y must contain the observations y_{mij} in the following order:

$$y_{111},y_{211},\ldots,y_{n_{11}11},y_{112},y_{212},\ldots,y_{n_{12}12},\ldots,y_{11l_1},\ldots,y_{n_{1l_1}1l_1},\ldots,y_{1ij},\ldots,y_{n_{ij}ij},\ldots,y_{1kl_k},\ldots,y_{n_{kl_k}kl_k}.$$

In words, the ordering is by group, and within each group is by subgroup, the members of each subgroup being in consecutive locations in Y.

2: N – INTEGER. *Input*

On entry: the total number of observations, n.

3: K – INTEGER. *Input*

On entry: the number of groups, k.

Constraint: K $\geq$ 2.

4: LSUB(K) – INTEGER array. *Input*

On entry: the number of subgroups within group i, l_i for $i = 1,2,\ldots,k$.

Constraint: LSUB(i) > 0 for $i = 1,2,\ldots,k$.

5: NOBS(L) – INTEGER array. *Input*

On entry: the numbers of observations in each subgroup, n_{ij}, in the following order:

$$n_{11},n_{12},\ldots,n_{1l_1},n_{21},\ldots,n_{2l_2},\ldots,n_{k1},\ldots,n_{kl_k}$$

Constraint: $n = \sum_{i=1}^{k}\sum_{j=1}^{l_i} n_{ij}$, that is N $= \sum_{i=1}^{l}$ NOBS(i) and NOBS(i) > 0 for $i = 1,2,\ldots,l$.

6: L – INTEGER. *Input*

On entry: the total number of subgroups, l.

Constraint: L $= \sum_{i=1}^{k}$ LSUB(i).

7: NGP(K) – INTEGER array. *Output*

On exit: the total number of observations in group i, $n_{i.}$, for $i = 1,2,\ldots,k$.

8: GBAR(K) – ***real*** array. *Output*

On exit: the mean for group i, $\bar{y}_{.i.}$, for $i = 1,2,\ldots,k$.

9: SGBAR(L) – ***real*** array. *Output*

On exit: the subgroup means, $\bar{y}_{.ij}$, in the following order:

$$\bar{y}_{.11},\bar{y}_{.12},\ldots,\bar{y}_{.1l_1},\bar{y}_{.21},\bar{y}_{.22},\ldots,\bar{y}_{.2l_2},\ldots,\bar{y}_{.k1},\bar{y}_{.k2},\ldots,\bar{y}_{.kl_k}.$$

10: GM – ***real***. *Output*

On exit: the grand mean, $\bar{y}_{...}$.

11: SS(4) – ***real*** array. *Output*

On exit: contains the sums of squares for the analysis of variance, as follows;

SS(1) = Between group sum of squares, SS_g,
SS(2) = Between subgroup within groups sum of squares, SS_{sg},
SS(3) = Residual sum of squares, SS_{res},
SS(4) = Corrected total sum of squares, SS_{tot}.

12: IDF(4) – INTEGER array. *Output*

On exit: contains the degrees of freedom attributable to each sum of squares in the analysis of variance, as follows;

IDF(1) = Degrees of freedom for between group sum of squares,
IDF(2) = Degrees of freedom for between subgroup within groups sum of squares,
IDF(3) = Degrees of freedom for residual sum of squares,
IDF(4) = Degrees of freedom for corrected total sum of squares.

13: F(2) – ***real*** array. *Output*

On exit: contains the mean square ratios, F_1 and F_2, for the between groups variation, and the between subgroups within groups variation, with respect to the residual, respectively.

14: FP(2) – ***real*** array. *Output*

On exit: contains the significances of the mean square ratios, p_1 and p_2 respectively.

15: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, K $\leq$ 1.

IFAIL = 2

On entry, LSUB(i) $\leq$ 0 for some $i = 1,2,...,k$.

IFAIL = 3

On entry, L $\neq \sum_{i=1}^{k}$ LSUB(i)

IFAIL = 4

On entry, NOBS(i) $\leq$ 0 for some $i = 1,2,...,l$.

IFAIL = 5

On entry, N $\neq \sum_{i=1}^{l}$ NOBS(i).

IFAIL = 6

The total corrected sum of squares is zero, indicating that all the data values are equal. The means returned are therefore all equal, and the sums of squares are zero. No assignments are made to IDF, F, and FP.

IFAIL = 7

The residual sum of squares is zero. This arises when either each subgroup contains exactly one observation, or the observations within each subgroup are equal. The means, sums of squares, and degrees of freedom are computed, but no assignments are made to F and FP.

7. Accuracy

The computations are believed to be stable.

8. Further Comments

The time taken by the routine increases approximately linearly with the total number of observations, *n*.

9. Example

The example below has two groups, the first of which consists of five subgroups, and the second of three subgroups. The number of observations in each subgroup are not equal. The data represent the percentage stretch in the length of samples of sack kraft drawn from consignments (subgroups) received over two years (groups). For details see Moore *et al.* [2].

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G04AGF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          K, LMAX, NMAX
      PARAMETER        (K=2,LMAX=8,NMAX=28)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             GM
      INTEGER          I, IFAIL, II, J, L, LI, N, NHI, NIJ, NLO, NSUB
*     .. Local Arrays ..
      real             F(2), FP(2), GBAR(K), SGBAR(LMAX), SS(4), Y(NMAX)
      INTEGER          IDF(4), LSUB(K), NGP(K), NOBS(LMAX)
*     .. External Subroutines ..
      EXTERNAL         G04AGF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G04AGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Data values'
      WRITE (NOUT,*)
      WRITE (NOUT,*) ' Group  Subgroup  Observations'
      LSUB(1) = 5
      LSUB(2) = 3
      L = LSUB(1) + LSUB(2)
      IF (L.LE.LMAX) THEN
         READ (NIN,*) (NOBS(I),I=1,L)
         N = 0
         DO 20 I = 1, L
            N = N + NOBS(I)
   20    CONTINUE
         IF (N.LE.NMAX) THEN
            READ (NIN,*) (Y(I),I=1,N)
            IFAIL = 1
            NSUB = 0
            NLO = 1
            DO 60 I = 1, K
               LI = LSUB(I)
               DO 40 J = 1, LI
                  NSUB = NSUB + 1
                  NIJ = NOBS(NSUB)
                  NHI = NLO + NIJ - 1
                  WRITE (NOUT,99999) I, J, (Y(II),II=NLO,NHI)
                  NLO = NLO + NIJ
   40          CONTINUE
   60       CONTINUE
*
```

```
            CALL G04AGF(Y,N,K,LSUB,NOBS,L,NGP,GBAR,SGBAR,GM,SS,IDF,F,FP,
     +                  IFAIL)
*
            IF (IFAIL.NE.0) THEN
               WRITE (NOUT,*)
               WRITE (NOUT,99997) 'Failed in G04AGF. IFAIL = ', IFAIL
            ELSE
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Subgroup means'
               WRITE (NOUT,*)
               WRITE (NOUT,*) '   Group  Subgroup  Mean'
               II = 0
               DO 100 I = 1, K
                  LI = LSUB(I)
                  DO 80 J = 1, LI
                     II = II + 1
                     WRITE (NOUT,99998) I, J, SGBAR(II)
   80             CONTINUE
  100          CONTINUE
               WRITE (NOUT,*)
               WRITE (NOUT,99996) '     Group 1 mean = ', GBAR(1),
     +           '    (', NGP(1), ' observations)'
               WRITE (NOUT,99996) '     Group 2 mean = ', GBAR(2),
     +           '    (', NGP(2), ' observations)'
               WRITE (NOUT,99996) '     Grand mean   = ', GM, '    (', N,
     +           ' observations)'
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Analysis of variance table'
               WRITE (NOUT,*)
               WRITE (NOUT,*)
     +           '   Source                SS     DF  F ratio  Sig'
               WRITE (NOUT,*)
               WRITE (NOUT,99995) 'Between groups          ', SS(1),
     +           IDF(1), F(1), FP(1)
               WRITE (NOUT,99995) 'Bet sbgps within gps    ', SS(2),
     +           IDF(2), F(2), FP(2)
               WRITE (NOUT,99995) 'Residual                ', SS(3),
     +           IDF(3)
               WRITE (NOUT,*)
               WRITE (NOUT,99995) 'Total                   ', SS(4),
     +           IDF(4)
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,I5,I9,4X,10F4.1)
99998 FORMAT (1X,I6,I8,F10.2)
99997 FORMAT (1X,A,I2)
99996 FORMAT (1X,A,F4.2,A,I2,A)
99995 FORMAT (1X,A,F5.3,I5,F7.2,F8.3)
      END
```

9.2. Program Data

```
G04AGF Example Program Data
 5 3 3 3 2 3 5 3
 2.1 2.4 2.0 2.0 2.0 2.4 2.1 2.2 2.4 2.2
 2.6 2.4 2.4 2.5 1.9 1.7 2.1 1.5 2.0 1.9
 1.7 1.9 1.9 1.9 2.0 2.1 2.3
```

9.3. Program Results

```
G04AGF Example Program Results

Data values

 Group  Subgroup  Observations
    1        1      2.1 2.4 2.0 2.0 2.0
    1        2      2.4 2.1 2.2
    1        3      2.4 2.2 2.6
    1        4      2.4 2.4 2.5
    1        5      1.9 1.7
    2        1      2.1 1.5 2.0
    2        2      1.9 1.7 1.9 1.9 1.9
    2        3      2.0 2.1 2.3

Subgroup means

   Group  Subgroup  Mean
     1        1      2.10
     1        2      2.23
     1        3      2.40
     1        4      2.43
     1        5      1.80
     2        1      1.87
     2        2      1.86
     2        3      2.13

    Group 1 mean = 2.21   (16 observations)
    Group 2 mean = 1.94   (11 observations)
    Grand mean   = 2.10   (27 observations)

Analysis of variance table

   Source                 SS      DF  F ratio  Sig

Between groups          0.475     1  16.15    0.001
Bet sbgps within gps    0.816     6   4.63    0.005
Residual                0.559    19

Total                   1.850    26
```

G04BBF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G04BBF computes the analysis of variance and treatment means and standard errors for a randomized block or completely randomized design.

2 Specification

```
      SUBROUTINE G04BBF(N, Y, IBLOCK, NT, IT, GMEAN, BMEAN, TMEAN,
     1                  TABLE, LDT, C, LDC, IREP, R, EF, TOL, IRDF, WK,
     2                  IFAIL)
      INTEGER           N, IBLOCK, NT, IT(*), LDT, LDC, IREP(NT), IRDF,
     1                  IFAIL
      real              Y(N), GMEAN, BMEAN(*), TMEAN(NT), TABLE(LDT,5),
     1                  C(LDC,NT), R(N), EF(NT), TOL, WK(NT*NT+NT)
```

3 Description

In a completely randomized design the experimental material is divided into a number of units, or plots, to which a treatment can be applied. In a randomized block design the units are grouped into blocks so that the variation within blocks is less than the variation between blocks. If every treatment is applied to one plot in each block it is a complete block design. If there are fewer plots per block than treatments then the design will be an incomplete block design and may be balanced or partially balanced.

For a completely randomized design, with t treatments and n_t plots per treatment, the linear model is:

$$y_{ij} = \mu + \tau_j + e_{ij}, \quad j = 1,2\ldots,t; i = 1,2\ldots,n_j,$$

where y_{ij} is the ith observation for the jth treatment, μ is the overall mean, τ_j is the effect of the jth treatment and e_{ij} is the random error term. For a randomised block design, with t treatments and b blocks of k plots, the linear model is:

$$y_{ij(l)} = \mu + \beta_i + \tau_l + e_{ij}, \quad i = 1,2\ldots,b; j = 1,2\ldots,k; l = 1,2,\ldots,t,$$

where β_i is the effect of the ith block and the $ij(l)$ notation indicates that the lth treatment is applied to the ith plot in the jth block.

The completely randomized design gives rise to a one-way analysis of variance. The treatments do not have to be equally replicated, i.e., do not have to occur the same number of times. First the overall mean, $\hat{\mu}$, is computed and subtracted from the observations to give, $y'_{ij} = y_{ij} - \hat{\mu}$. The estimated treatment effects, $\hat{\tau}_j$ are then computed as the treatment means of the mean adjusted observations, y'_{ij}, and the treatment sum of squares can be computed from the sum of squares of the treatment totals of the y'_{ij} divided by the number of observations per treatment total, n_j. The final residuals are computed as $r_{ij} = y'_{ij} - \hat{\tau}_j$, and, from the residuals, the residual sum of squares is calculated.

For the randomised block design the mean is computed and subtracted from the observations to give, $y'_{ij(l)} = y_{ij(l)} - \hat{\mu}$. The estimated block effects, ignoring treatment effects, $\hat{\beta}_i$, are then computed using the block means of the $y'_{ij(l)}$ and the unadjusted sum of squares computed as the sum of squared block totals for the $y'_{ij(l)}$ divided by number of plots per block, k. The block adjusted observations are then computed as $y''_{ij(l)} = y'_i j(l) = \hat{\beta}_i$. In the case of the complete block design, with the same replication for each treatment within each block, the blocks and treatments are orthogonal,m and so the treatment effects are estimated as the treatment means of the block adjusted observations, $y''_{ij(l)}$. The treatment sum of squares is computed as the sum of squared treatment totals of the $y''_{ij(l)}$ divided by the number of replicates to the treatments, $r = bk/t$. Finally the residuals, and hence the residual sum of squares, are given by $r_{ij(l)} = y''_{ij(l)} - \hat{\tau}_l$.

For a design without the same replication for each treatment within each block the treatments and the blocks will not be orthogonal, so the treatments adjusted for blocks need to be computed. The adjusted treatment effects are found as the solution to the equations:

$$(R - NN^T/k)\hat{\tau} = q$$

where q is the vector of the treatment totals for block adjusted observations, $y''_{ij(l)}$, R is a diagonal matrix with R_{ll} equal to the number of times the lth treatment is replicated, and N is the t by b incidence matrix, with $N_{l,j}$ equal to the number of times treatment l occurs in block j. The solution to the equations can be written as:

$$\hat{\tau} = \Omega q$$

where Ω is a generalized inverse of $(R - NN^T/k)$. The solution is found from the eigenvalue decomposition of $(R - NN^T/k)$. The residuals are first calculated by subtracting the estimated treatment effects from the block adjusted observations to give $r'_{ij(l)} = y''_{ij(l)} - \hat{\tau}_l$. However, since only the unadjusted block effects have been removed and blocks and treatments are not orthogonal, the block means of the $r'_{ij(l)}$ have to give the correct residuals, $r_{ij(l)}$ and residual sum of squares.

The mean squares are computed as the sum of squares divided by the degrees of freedom. The degrees of freedom for the unadjusted blocks is $b - 1$, for the completely randomised and the complete block designs the degrees of freedom for the treatments is $t - 1$. In the general case the degrees of freedom for treatments is the rank of the matrix Ω. The F-statistic given by the ratio of the treatment mean square to the residual mean square tests the hypothesis:

$$H_0 : \tau_1 = \tau_2 = \ldots = \tau_t = 0.$$

The standard errors for the difference in treatment effects, or treatment means, for the completely randomized or the complete block designs, are given by:

$$se(\tau_j - \tau_{j*}) = \left(\frac{1}{n_j} + \frac{1}{n_{j*}}\right) s^2$$

where s^2 is the residual mean square and $n_j = n_{j*} = b$ in the complete block design. In the general case the variances of the treatment effects are given by:

$$\mathrm{var}(\tau) = \Omega s^2$$

from which the appropriate standard errors of the difference between treatment effects or the difference between adjusted means can be calculated.

In the complete block design all the information on the treatment effects is given by the within block analysis. In other designs there may be a loss of information due to the non-orthogonality of treatments and blocks. The efficiency of the within block analysis in these cases is given by the (canonical) efficiency factors, these are the non-zero eigenvalues of the matrix $(R - NN^T/k)$, divided by the number of replicates in the case of equal replication, or by the mean of the number of replicates in the unequally replicated case, see John [3]. If more than one eigenvalue is zero then the design is said to be disconnected and some treatments can only be compared using a between block analysis.

4 References

[1] Cochran W G and Cox G M (1957) *Experimental Designs* Wiley

[2] Davis O L (1978) *The Design and Analysis of Industrial Experiments* Longman

[3] John J A (1987) *Cyclic Designs* Chapman and Hall

[4] John J A and Quenouille M H (1977) *Experiments: Design and Analysis* Griffin

[5] Searle S R (1971) *Linear Models* Wiley

5 Parameters

1: N — INTEGER *Input*

On entry: the number of observations.

Constraints: N $\geq$ 2 and if abs(IBLOCK) $\geq$ 2, N must be a multiple of abs(IBLOCK).

2: Y(N) — ***real*** array *Input*

On entry: the observations in the order as described by IBLOCK and NT.

3: IBLOCK — INTEGER *Input*

On entry: IBLOCK indicates the block structure. If abs(IBLOCK) $\leq$ 1 then there are no blocks, i.e., it is a completely randomized design. If IBLOCK $\geq$ 2 then there are IBLOCK blocks and the data should be input by blocks, i.e., Y must contain the observations for block 1 followed by the observations for block 2 etc.. If IBLOCK ≤ -2 then there are abs(IBLOCK) blocks and the data is input in parallel with respect to blocks, i.e., Y(1) must contain the first observation for block 1, Y(2) must contain the first observation for block 2 ... Y(abs(IBLOCK)) must contain the first observation for block abs(IBLOCK),Y(abs(IBLOCK+1)) must contain the second observation for block 1 etc..

4: NT — INTEGER *Input*

On entry: the number of treatments. If only blocks are required in the analysis then set NT = 1.

Constraint: if abs(IBLOCK) $\geq$ 2, NT $\geq$ 1, otherwise NT $\geq$ 2.

5: IT(*) — INTEGER array *Input*

Note: the dimension of the array IT must be at least N if NT $\geq$ 2, and 1 otherwise.

On entry: IT(i) indicates which of the NT treatments plot i received, for $i = 1, 2, \ldots, N$. If NT = 1, IT is not referenced.

Constraint: $1 \leq$ IT(i) $\leq$ NT , for $i = 1, 2, \ldots,$ N.

6: GMEAN — ***real*** *Output*

On exit: the grand mean, $\hat{\mu}$.

7: BMEAN(*) — ***real*** array *Output*

Note: the dimension of the array BMEAN must be at least max(1,abs(IBLOCK)).

On exit: if abs(IBLOCK) $\geq$ 2, BMEAN(j) contains the mean for the jth block, $\hat{\beta}_j$ for $j = 1, 2, \ldots, b$.

8: TMEAN(NT) — ***real*** array *Output*

On exit: if NT $\geq$ 2, TMEAN(l) contains the (adjusted) mean for the lth treatment, $\hat{\mu}^* + \hat{\tau}_l$, for $l = 1, 2, \ldots, t$, where $\hat{\mu}^*$ is the mean of the treatment adjusted observations, $y_{ij(l)} - \hat{\tau}_l$.

9: TABLE(LDT,5) — ***real*** array *Output*

On exit: the analysis of variance table. Column 1 contains the degrees of freedom, column 2 the sum of squares, and where appropriate, column 3 the mean squares, column 4 the F-statistic and column 5 the significance level of the F-statistic. Row 1 is for Blocks, row 2 for Treatments, row 3 for Residual and row 4 for Total. Mean squares are computed for all but the Total row; F-statistics and significance are computed for Treatments and Blocks, if present. Any unfilled cells are set to zero.

10: LDT — INTEGER *Input*

On entry: the first dimension of the array TABLE as declared in the (sub)program from which G04BBF is called.

Constraint: LDT $\geq$ 4.

11: C(LDC,NT) — ***real*** array *Output*

On exit: if NT $\geq$ 2, the upper triangular part of C contains the variance-covariance matrix of the treatment effects, the strictly lower triangular part contains the standard errors of the difference between two treatment effects (means), i.e., $C(i,j)$ contains the covariance of treatment i and j if $j \geq i$ and the standard error of the difference between treatment i and j if $j < i$ for $i = 1,2,\ldots,t$; $j = 1,2,\ldots,t$.

12: LDC — INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which G04BBF is called.

Constraint: LDC $\geq$ NT.

13: IREP(NT) — INTEGER array *Output*

On exit: if NT $\geq$ 2, the treatment replications, R_{ll}, for $l = 1,2,\ldots,$NT.

14: R(N) — ***real*** array *Output*

On exit: the residuals, r_i, for $i = 1,2,\ldots$N.

15: EF(NT) — ***real*** array *Output*

On exit: if NT $\geq$ 2, the canonical efficiency factors.

16: TOL — ***real*** *Input*

On entry: the tolerance value used to check for zero eigenvalues of the matrix Ω. If TOL = 0.0 a default value of 10^{-5} is used.

Constraint: TOL $\geq$ 0.0.

17: IRDF — INTEGER *Input*

On entry: an adjustment to the degrees of freedom for the residual and total. If IRDF $\geq$ 1 the degrees of freedom for the total is set to N − IRDF and the residual degrees of freedom adjusted accordingly. If IRDF = 0, the total degrees of freedom for the total is set to N − 1, as usual.

Constraint: IRDF $\geq$ 0.

18: WK(NT*NT+NT) — ***real*** array *Workspace*

19: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL $\neq$ 0 on exit, users are recommended to set IFAIL to −1 before entry. **It is then essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings specified by the routine:

IFAIL = 1

On entry, N < 2,
or NT $\leq$ 0,
or NT = 1 and abs(IBLOCK) $\leq$ 1,

or LDT < 4,
or LDC $<$ NT,
or TOL < 0.0,
or IRDF < 0.

IFAIL $= 2$

On entry, abs(IBLOCK) ≥ 2 and N is not a multiple of abs(IBLOCK).

IFAIL $= 3$

On entry, IT(i) < 1 or IT(i) $>$ NT for some i when NT ≥ 2,
or no value of IT $= j$ for some $j = 1, 2, \ldots,$ NT, when NT ≥ 2.

IFAIL $= 4$

On entry, the values of Y are constant.

IFAIL $= 5$

A computed standard error is zero due to rounding errors, or the eigenvalue computation failed to converge. Both are unlikely error exits.

IFAIL $= 6$

The treatments are totally confounded with blocks, so the treatment sum of squares and degrees of freedom are zero. The analysis of variance table is not computed, except for block and total sums of squares and degrees of freedom.

IFAIL $= 7$

The residual degrees of freedom or the residual sum of squares are zero, columns 3, 4 and 5 of the analysis of variance table will not be computed and the matrix of standard errors and covariances, C, will not be scaled by s or s^2.

IFAIL $= 8$

The design is disconnected; the standard errors may not be valid. The design may be nested.

7 Accuracy

The algorithm used by this routine, described in Section 3, achieves greater accuracy than the traditional algorithms based on the subtraction of sums of squares.

8 Further Comments

To estimate missing values the Healy and Westmacott procedure or its derivatives may be used, see [4]. This is an iterative procedure in which estimates of the missing values are adjusted by subtracting the corresponding values of the residuals. The new estimates are then used in the analysis of variance. This process is repeated until convergence. A suitable initial value may be the grand mean $\hat{\mu}$. When using this procedure IRDF should be set to the number of missing values plus one to obtain the correct degrees of freedom for the residual sum of squares.

For designs such as Graeco-Latin squares one or more of the blocking factors has to be removed in a preliminary analysis before the final analysis using calls to G04BBF or G04BCF. The residuals from the preliminary analysis are then input to G04BBF. In these cases IRDF should be set to the difference between N and the residual degrees of freedom from preliminary analysis. Care should be taken when using this approach as there is no check on the orthogonality of the two analyses.

For analysis of covariance the residuals are obtained from an analysis of variance of both the response variable and the covariates. The residuals from the response variable are then regressed on the residuals from the covariates using, say, G02CBF or G02DAF. The results from those routines can be used to test for the significance of the covariates. To test the significance of the treatment effects after fitting the covariate, the residual sum of squares from the regression should be compared with the residual sum of squares obtained from the equivalent regression but using the residuals from fitting blocks only.

9 Example

The data, given by John and Quenouille [4], are for a balanced incomplete block design with 10 blocks and 6 treatments and with 3 plots per block. The observations are the degree of pain experienced and the treatments are penicillin of different potency. The data is input and the analysis of variance table and treatment means are printed.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G04BBF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NTMAX, NBMAX, TMAX
      PARAMETER        (NMAX=30,NTMAX=6,NBMAX=10,TMAX=4)
*     .. Local Scalars ..
      real             GMEAN, TOL
      INTEGER          I, IFAIL, IRDF, J, N, NBLOCK, NT
*     .. Local Arrays ..
      real             BMEAN(NBMAX), C(NTMAX,NTMAX), EF(NTMAX), R(NMAX),
     +                 TABLE(TMAX,5), TMEAN(NTMAX),
     +                 WK(NTMAX*NTMAX+NTMAX), Y(NMAX)
      INTEGER          IREP(NTMAX), IT(NMAX)
*     .. External Subroutines ..
      EXTERNAL         G04BBF
*     .. Executable Statements ..
      WRITE (NOUT,FMT=*) 'G04BBF Example Program Results'
*     Skip heading in data file
      READ (NIN,FMT=*)
      READ (NIN,FMT=*) N, NT, NBLOCK
      IF (N.LE.NMAX) THEN
         READ (NIN,FMT=*) (Y(I),I=1,N)
         READ (NIN,FMT=*) (IT(I),I=1,N)
         TOL = 0.000005e0
         IRDF = 0
         IFAIL = -1
*
         CALL G04BBF(N,Y,NBLOCK,NT,IT,GMEAN,BMEAN,TMEAN,TABLE,TMAX,C,
     +               NTMAX,IREP,R,EF,TOL,IRDF,WK,IFAIL)
*
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*) ' ANOVA table'
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*)
     +     ' Source          df          SS          MS          F',
     +     '          Prob'
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=99998) ' Blocks       ', (TABLE(1,J),J=1,5)
         WRITE (NOUT,FMT=99998) ' Treatments   ', (TABLE(2,J),J=1,5)
         WRITE (NOUT,FMT=99998) ' Residual     ', (TABLE(3,J),J=1,3)
         WRITE (NOUT,FMT=99998) ' Total        ', (TABLE(4,J),J=1,2)
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*) ' Efficiency Factors'
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=99999) (EF(I),I=1,NT)
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=99997) '  Grand Mean', GMEAN
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*) ' Treatment Means'
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=99999) (TMEAN(I),I=1,NT)
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*)
     +     ' Standard errors of differences between means'
         WRITE (NOUT,FMT=*)
         DO 20 I = 2, NT
            WRITE (NOUT,FMT=99999) (C(I,J),J=1,I-1)
   20    CONTINUE
      END IF
      STOP
*
99999 FORMAT (8F10.2)
99998 FORMAT (A,3X,F3.0,2X,3(F10.2,2X),F9.4)
99997 FORMAT (A,F10.2)
      END
```

9.2 Program Data

```
G04BBF Example Program Data
30 6 10                   : N, NT, IBLOCK
1 5 4
5 10 6
2 9 3
4 8 6
2 4 7
6 7 5
5 7 2
7 2 4
8 4 2
10 8 7                    : Y
1 2 3
1 2 4
1 3 5
1 4 6
1 5 6
2 3 6
2 4 5
2 5 6
3 4 5
3 4 6                     : IT
```

9.3 Program Results

```
 G04BBF Example Program Results

  ANOVA table

   Source        df          SS          MS          F       Prob

 Blocks          9.       60.00        6.67       4.79     0.0039
 Treatments      5.      101.78       20.36      14.62     0.0000
 Residual      15.       20.89        1.39
 Total         29.      182.67

  Efficiency Factors

      0.00      0.80      0.80      0.80      0.80      0.80

  Grand Mean        5.33

  Treatment Means

      2.50      7.25      8.08      5.92      2.92      5.33

  Standard errors of differences between means

      0.83
      0.83      0.83
      0.83      0.83      0.83
      0.83      0.83      0.83      0.83
      0.83      0.83      0.83      0.83      0.83
```

G04BCF – NAG Fortran Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G04BCF computes the analysis of variance for a general row and column design together with the treatment means and standard errors.

2 Specification

```
      SUBROUTINE G04BCF(NREP, NROW, NCOL, Y, NT, IT, GMEAN, TMEAN,
     1                  TABLE, LDT, C, LDC, IREP, RPMEAN, RMEAN, CMEAN,
     2                  R, EF, TOL, IRDF, WK, IFAIL)
      INTEGER           NREP, NROW, NCOL, NT, IT(*), LDT, LDC, IREP(NT),
     1                  IRDF, IFAIL
      real              Y(NREP*NROW*NCOL), GMEAN, TMEAN(NT),
     1                  TABLE(LDT,5), C(LDC,NT), RPMEAN(NREP),
     2                  RMEAN(NREP*NROW), CMEAN(NREP*NCOL),
     3                  R(NREP*NROW*NCOL), EF(NT), TOL, WK(3*NT)
```

3 Description

In a row and column design the experimental material can be characterized by a two-way classification, nominally called rows and columns. Each experimental unit can be considered as being located in a particular row and column. It is assumed that all rows are of the same length and all columns are of the same length. Sets of equal numbers of rows/columns can be grouped together to form replicates, sometimes known as squares or rectangles, as appropriate.

If for a replicate, the number of rows, the number of columns and the number of treatments are equal and every treatment occurs once in each row and each column then the design is a Latin square. If this is not the case the treatments will be non-orthogonal to rows and columns. For example in the case of a lattice square each treatment occurs only once in each square.

For a row and column design, with t treatments in r rows and c columns and b replicates or squares with $n = brc$ observations the linear model is:

$$y_{ijk(l)} = \mu + \beta_i + \rho_j + \gamma_k + \tau_l + e_{ijk}$$

$i = 1,2\ldots,b$; $j = 1,2,\ldots,r$; $k = 1,2\ldots,c$; $l = 1,2,\ldots,t$, where β_i is the effect of the ith replicate, ρ_j is the effect of the jth row, γ_k is the effect of the kth column and the $ijk(l)$ notation indicates that the lth treatment is applied to the unit in row j, column k of replicate i.

To compute the analysis of variance for a row and column design the mean is computed and subtracted from the observations to give, $y'_{ijk(l)} = y_{ijk(l)} - \hat{\mu}$. Since the replicates, rows and columns are orthogonal the estimated effects, ignoring treatment effects, $\hat{\beta}_i$, $\hat{\rho}_j$, $\hat{\gamma}_k$, can be computed using the appropriate means of the $y'_{ijk(l)}$, and the unadjusted sum of squares computed as the appropriate sum of squared totals for the $y'_{ijk(l)}$ divided by number of units per total. The observations adjusted for replicates, rows and columns can then be computed by subtracting the estimated effects from $y'_{ijk(l)}$ to give $y''_{ijk(l)}$.

In the case of a Latin square design the treatments are orthogonal to replicates, rows and columns and so the treatment effects, $\hat{\tau}_l$, can be estimated as the treatment means of the adjusted observations, $y''_{ijk(l)}$. The treatment sum of squares is computed as the sum of squared treatment totals of the $y''_{ij(l)}$ divided by the number of times each treatment is replicated. Finally the residuals, and hence the residual sum of squares, are given by, $r_{ij(l)} = y''_{ij(l)} - \hat{\tau}_l$.

For a design which is not orthogonal, for example a lattice square or an incomplete latin square, the treatment effects adjusted for replicates, rows and columns need to be computed. The adjusted treatment effects are found as the solution to the equations:

$$A\hat{\tau} = (R - N_b N_b^T/(rc) - N_r N_r^T/(bc) - N_c N_c^T/(br))\hat{\tau} = q$$

where q is the vector of the treatment totals of the observations adjusted for replicates, rows and columns, $y''_{ijk(l)}$, R is a diagonal matrix with R_{ll} equal to the number of times the lth treatment is replicated, and N_b is the t by b incidence matrix, with $N_{l,i}$ equal to the number of times treatment l occurs in replicate i, with N_r and N_c being similarly defined for rows and columns. The solution to the equations can be written as:

$$\hat{\tau} = \Omega q$$

where, Ω is a generalized inverse of A. The solution is found from the eigenvalue decomposition of A. The residuals are first calculated by subtracting the estimated adjusted treatment effects from the adjusted observations to give $r'_{ij(l)} = y''_{ij(l)} - \hat{\tau}_l$. However, since only the unadjusted replicate, row and column effects have been removed and they are not orthogonal to treatments, the replicate, row and column means of the $r'_{ij(l)}$ have to be subtracted to give the correct residuals, $r_{ij(l)}$ and residual sum of squares.

Given the sums of squares, the mean squares are computed as the sums of squares divided by the degrees of freedom. The degrees of freedom for the unadjusted replicates, rows and columns are $b-1$, $r-1$ and $c-1$ respectively and for the Latin square designs the degrees of freedom for the treatments is $t-1$. In the general case the degrees of freedom for treatments is the rank of the matrix Ω. The F-statistic given by the ratio of the treatment mean square to the residual mean square tests the hypothesis:

$$H_0 : \tau_1 = \tau_2 = \ldots = \tau_t = 0.$$

The standard errors for the difference in treatment effects, or treatment means, for Latin square designs, are given by:

$$se(\hat{\tau}_j - \hat{\tau}_{j*}) = \sqrt{2s^2/(bt)}$$

where s^2 is the residual mean square. In the general case the variances of the treatment effects are given by:

$$\text{Var}(\hat{\tau}) = \Omega s^2$$

from which the appropriate standard errors of the difference between treatment effects or the difference between adjusted means can be calculated.

The analysis of a row-column design can be considered as consisting of different strata: the replicate stratum, the rows within replicate and the columns within replicate strata and the units stratum. In the Latin square design all the information on the treatment effects is given at the units stratum. In other designs there may be a loss of information due to the non-orthogonality of treatments and replicates, rows and columns and information on treatments may be available in higher strata. The efficiency of the estimation at the units stratum is given by the (canonical) efficiency factors, these are the non-zero eigenvalues of the matrix, A, divided by the number of replicates in the case of equal replication, or by the mean of the number of replicates in the unequally replicated case, see John [3]. If more than one eigenvalue is zero then the design is said to be disconnected and information on some treatment comparisons can only be obtained from higher strata.

4 References

[1] Cochran W G and Cox G M (1957) *Experimental Designs* Wiley

[2] Davis O L (1978) *The Design and Analysis of Industrial Experiments* Longman

[3] John J A (1987) *Cyclic Designs* Chapman and Hall

[4] John J A and Quenouille M H (1977) *Experiments: Design and Analysis* Griffin

[5] Searle S R (1971) *Linear Models* Wiley

5 Parameters

1: NREP — INTEGER *Input*

On entry: the number of replicates, b.

Constraint: NREP ≥ 1.

2: NROW — INTEGER *Input*

On entry: the number of rows per replicate, r.

Constraint: NROW ≥ 2.

3: NCOL — INTEGER *Input*

On entry: the number of columns per replicate, c.

Constraint: NCOL ≥ 2.

4: Y(NREP*NROW*NCOL) — ***real*** array *Input*

On entry: the $n = brc$ observations ordered by columns within rows within replicates. That is Y$(rc(i-1)+r(j-1)+k)$ contains the observation from the k column of the jth row of the ith replicate for $i = 1,2,\ldots,b$; $j = 1,2\ldots,r$; $k = 1,2,\ldots,c$.

5: NT — INTEGER *Input*

On entry: the number of treatments. If only replicates, rows and columns are required in the analysis then set NT $= 1$.

Constraint: NT ≥ 1.

6: IT(*) — INTEGER array *Input*

Note: the dimension of the array IT must be at least NREP*NROW*NCOL if NT > 1, and 1 otherwise.

On entry: if NT > 1, T(i) indicates which of the NT treatments unit i received, for $i = 1,2,\ldots,n$. If NT $= 1$, IT is not referenced.

Constraint: if NT ≥ 2, $1 \leq$ IT$(i) \leq$ NT, for $i = 1,2,\ldots,n$.

7: GMEAN — ***real*** *Output*

On exit: the grand mean, $\hat{\mu}$.

8: TMEAN(NT) — ***real*** array *Output*

On exit: if NT ≥ 2, TMEAN(l) contains the (adjusted) mean for the lth treatment, $\hat{\mu}^* + \hat{\tau}_l$, for $l = 1,2,\ldots,t$, where $\hat{\mu}^*$ is the mean of the treatment adjusted observations $y_{ijk(l)} - \hat{\tau}_l$. Otherwise TMEAN is not referenced.

9: TABLE(LDT,5) — ***real*** array *Output*

On exit: the analysis of variance table. Column 1 contains the degrees of freedom, column 2 the sum of squares, and where appropriate, column 3 the mean squares, column 4 the F-statistic and column 5 the significance level of the F-statistic. Row 1 is for replicates, row 2 for rows, row 3 for columns, row 4 for treatments (if NT > 1), row 5 for residual and row 6 for total. Mean squares are computed for all but the total row, F-statistics and significance are computed for treatments, replicates, rows and columns. Any unfilled cells are set to zero.

10: LDT — INTEGER *Input*

On entry: the first dimension of the array TABLE as declared in the (sub)program from which G04BCF is called.

Constraint: LDT ≥ 6.

11: C(LDC,NT) — ***real*** array *Output*

On exit: the upper triangular part of C contains the variance-covariance matrix of the treatment effects, the strictly lower triangular part contains the standard errors of the difference between two treatment effects (means), i.e., C(i,j) contains the covariance of treatment i and j if $j \geq i$ and the standard error of the difference between treatment i and j if $j < i$ for $i = 1,2,\ldots,t$; $j = 1,2,\ldots,t$.

12: LDC — INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which G04BCF is called.

Constraint: LDC $\geq$ NT.

13: IREP(NT) — INTEGER array *Output*

On exit: if NT > 1, the treatment replications, R_{ll}, for $l = 1,2,\ldots,$NT. Otherwise IREP is not referenced.

14: RPMEAN(NREP) — ***real*** array *Output*

On exit: if NREP > 1, RPMEAN(i) contains the mean for the ith replicate, $\hat{\mu}+\hat{\beta}_i$ for $i = 1,2,\ldots,b$. Otherwise RPMEAN is not referenced.

15: RMEAN(NREP*NROW) — ***real*** array *Output*

On exit: RMEAN(j) contains the mean for the jth row, $\hat{\mu} + \hat{\rho}_i$ for $j = 1,2,\ldots,r$.

16: CMEAN(NREP*NCOL) — ***real*** array *Output*

On exit: CMEAN(k) contains the mean for the kth column, $\hat{\mu} + \hat{\gamma}_k$ for $k = 1,2,\ldots,c$.

17: R(NREP*NROW*NCOL) — ***real*** array *Output*

On exit: the residuals, r_i for $i = 1,2,\ldots,n$.

18: EF(NT) — ***real*** array *Output*

On exit: if NT ≥ 2, the canonical efficiency factors. Otherwise EF is not referenced.

19: TOL — ***real*** *Input*

On entry: the tolerance value used to check for zero eigenvalues of the matrix Ω. If TOL $= 0.0$ a default value of 0.00001 is used.

Constraint: TOL ≥ 0.0

20: IRDF — INTEGER *Input*

On entry: an adjustment to the degrees of freedom for the residual and total. If IRDF ≥ 1 the degrees of freedom for the total is set to $n -$ IRDF and the residual degrees of freedom adjusted accordingly. If IRDF $= 0$, the total degrees of freedom for the total is set to $n - 1$, as usual.

Constraint: IRDF ≥ 0.

21: WK(3*NT) — ***real*** array *Workspace*

22: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL $= 0$ unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL $\neq 0$ on exit, users are recommended to set IFAIL to -1 before entry. **It is then essential to test the value of IFAIL on exit.**

6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings specified by the routine:

IFAIL= 1

On entry, NREP < 1,
or NROW < 2,
or NCOL < 2,
or NT < 1,
or LDT < 6,
or LDC $<$ NT,
or TOL < 0.0,
or IRDF < 0.

IFAIL=2

On entry, IT(i) < 1 or IT(i) $>$ NT for some i when NT ≥ 2,
or no value of IT $= j$ for some $j = 1, 2, \ldots,$ NT, when NT ≥ 2.

IFAIL= 3

On entry, the values of Y are constant.

IFAIL= 4

A computed standard error is zero due to rounding errors, or the eigenvalue computation failed to converge. Both are unlikely error exits.

IFAIL=5

The treatments are totally confounded with replicates, rows and columns, so the treatment sum of squares and degrees of freedom are zero. The analysis of variance table is not computed, except for replicate, row, column and total sums of squares and degrees of freedom.

IFAIL=6

The residual degrees of freedom or the residual sum of squares are zero, columns 3, 4 and 5 of the analysis of variance table will not be computed and the matrix of standard errors and covariances, C, will not be scaled by s or s^2.

IFAIL=7

The design is disconnected, the standard errors may not be valid. The design may have a nested structure.

7 Accuracy

The algorithm used in this routine, described in Section 3, achieves greater accuracy than the traditional algorithms based on the subtraction of sums of squares.

8 Further Comments

To estimate missing values the Healy and Westmacott procedure or its derivatives may be used, see [4]. This is an iterative procedure in which estimates of the missing values are adjusted by subtracting the corresponding values of the residuals. The new estimates are then used in the analysis of variance. This process is repeated until convergence. A suitable initial value may be the grand mean. When using this procedure IRDF should be set to the number of missing values plus one to obtain the correct degrees of freedom for the residual sum of squares.

For analysis of covariance the residuals are obtained from an analysis of variance of both the response variable and the covariates. The residuals from the response variable are then regressed on the residuals from the covariates using, say, G02CBF or G02DAF. The results from those routines can be used to test for the significance of the covariates. To test the significance of the treatment effects after fitting the covariate, the residual sum of squares from the regression should be compared with the residual sum of squares obtained from the equivalent regression but using the residuals from fitting replicates, rows and columns only.

9 Example

The data for a 5 × 5 Latin square is input and the ANOVA and treatment means computed and printed. Since the design is orthogonal only one standard error need be printed

9.1 Example Text

Note: the listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G04BCF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NTMAX, NBMAX
      PARAMETER        (NMAX=25,NTMAX=5,NBMAX=5)
*     .. Local Scalars ..
      real             GMEAN
      INTEGER          I, IFAIL, J, N, NCOL, NREP, NROW, NT
*     .. Local Arrays ..
      real             C(NTMAX,NTMAX), CMEAN(NBMAX), EF(NTMAX), R(NMAX),
     +                 RMEAN(NBMAX), RPMEAN(NBMAX), TABLE(6,5),
     +                 TMEAN(NTMAX), WK(3*NTMAX), Y(NMAX)
      INTEGER          IREP(NTMAX), IT(NMAX)
*     .. External Subroutines ..
      EXTERNAL         G04BCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G04BCF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) NREP, NROW, NCOL, NT
      IF (NROW.LE.NBMAX .AND. NCOL.LE.NBMAX .AND. NT.LE.NTMAX) THEN
         N = NREP*NROW*NCOL
         READ (NIN,*) (Y(I),I=1,N)
         READ (NIN,*) (IT(I),I=1,N)
         IFAIL = -1
*
         CALL G04BCF(NREP,NROW,NCOL,Y,NT,IT,GMEAN,TMEAN,TABLE,6,C,NTMAX,
     +               IREP,RPMEAN,RMEAN,CMEAN,R,EF,0.00001e0,0,WK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' ANOVA TABLE'
         WRITE (NOUT,*)
         IF (NREP.GT.1) THEN
            WRITE (NOUT,99998) ' Reps         ', (TABLE(1,J),J=1,5)
         END IF
         WRITE (NOUT,99998) ' Rows         ', (TABLE(2,J),J=1,5)
         WRITE (NOUT,99998) ' Columns      ', (TABLE(3,J),J=1,5)
         WRITE (NOUT,99998) ' Treatments   ', (TABLE(4,J),J=1,5)
         WRITE (NOUT,99998) ' Residual    ', (TABLE(5,J),J=1,3)
         WRITE (NOUT,99998) ' Total       ', (TABLE(NOUT,J),J=1,2)
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Treatment means'
         WRITE (NOUT,*)
         WRITE (NOUT,99999) (TMEAN(I),I=1,NT)
         WRITE (NOUT,*)
         WRITE (NOUT,99997) ' S.E. of difference (orthogonal design) = '
     +     , C(2,1)
      END IF
      STOP
*
99999 FORMAT (8F10.4)
99998 FORMAT (A,F3.0,2X,3(F10.4,2X),F8.4)
99997 FORMAT (A,F10.4)
      END
```

9.2 Example Data

```
G04BCF Example Program Data

1 5 5 5

6.67  7.15  8.29  8.95  9.62
5.40  4.77  5.40  7.54  6.93
7.32  8.53  8.50  9.99  9.68
4.92  5.00  7.29  7.85  7.08
4.88  6.16  7.83  5.38  8.51

5  4  1  3  2
2  5  4  1  3
3  2  5  4  1
1  3  2  5  4
4  1  3  2  5
```

9.3 Example Results

```
 G04BCF Example Program Results

  ANOVA TABLE

 Rows          4.      29.4231      7.3558       9.0266    0.0013
 Columns       4.      22.9950      5.7487       7.0545    0.0037
 Treatments    4.       0.5423      0.1356       0.1664    0.9514
 Residual     12.       9.7788      0.8149
 Total        24.      62.7392

  Treatment means

    7.3180    7.2440    7.2060    6.9000    7.2600

 S.E. of difference (orthogonal design) =     0.5709
```

G04CAF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G04CAF computes an analysis of variance table and treatment means for a complete factorial design.

2 Specification

```
      SUBROUTINE G04CAF(N, Y, NFAC, LFAC, NBLOCK, INTER, IRDF, MTERM,
     1                  TABLE, ITOTAL, TMEAN, MAXT, E, IMEAN, SEMEAN,
     2                  BMEAN, R, IWK, IFAIL)
      INTEGER           N, NFAC, LFAC(NFAC), NBLOCK, INTER, IRDF, MTERM,
     1                  ITOTAL, MAXT, IMEAN(MTERM), IWK(N+3*NFAC), IFAIL
      real              Y(N), TABLE(MTERM,5), TMEAN(MAXT), E(MAXT),
     1                  SEMEAN(MTERM), BMEAN(NBLOCK+1), R(N)
```

3 Description

An experiment consists of a collection of units, or plots, to which a number of treatments are applied. In a factorial experiment the effects of several different sets of conditions are compared, e.g. three different temperatures, T_1, T_2 and T_3, and two different pressures, P_1 and P_2. The conditions are known as factors and the different values the conditions take are known as levels. In a factorial experiment the experimental treatments are the combinations of all the different levels of all factors. e.g.,

$$T_1P_1,\quad T_2P_1,\quad T_3P_1$$

$$T_1P_2,\quad T_2P_2,\quad T_3P_2$$

The effect of a factor averaged over all other factors is known as a main effect and the effect of a combination of some of the factors averaged over all other factors is known as an interaction. This can be represented by a linear model. In the above example if the response was y_{ijk} for the kth replicate of the ith level of T and the jth level of P the linear model would be:

$$y_{ijk} = \mu + t_i + p_j + \gamma_{ij} + e_{ijk}$$

where μ is the overall mean, t_i is the main effect of T, p_j is the main effect of P, γ_{ij} is the $T \times P$ interaction and e_{ijk} is the random error term. In order to find unique estimates constraints are placed on the parameters estimates. For the example here these are:

$$\sum_{i=1}^{3} \hat{t}_i = 0,$$

$$\sum_{j=1}^{2} \hat{p}_j = 0,$$

$$\sum_{i=1}^{3} \hat{\gamma}_{ij} = 0 \text{ for } j = 1,2 \text{ and}$$

$$\sum_{j=1}^{2} \hat{\gamma}_{ij} = 0 \text{ for } i = 1,2,3,$$

where $\hat{}$ denotes the estimate.

If there is variation in the experimental conditions, e.g. in an experiment on the production of a material different batches of raw material may be used, or the experiment may be carried out on different days, then plots that are similar are grouped together into blocks. For a balanced complete factorial experiment all the treatment combinations occur the same number of times in each block.

G04CAF computes the analysis of variance (ANOVA) table by sequentially computing the totals and means for an effect from the residuals computed when previous effects have been removed. The effect sum of squares is the sum of squared totals divided by the number of observations per total. The means are then subtracted from the residuals to compute a new set of residuals. At the same time the means for the original data are computed. When all effects are removed the residual sum of squares is computed from the residuals. Given the sums of squares an ANOVA table is then computed along with standard errors for the difference in treatment means.

The data for G04CAF has to be in standard order given by the order of the factors. Let there be k factors, $f_1, f_2, \ldots, f_k$ in that order with levels $l_1, l_2, \ldots, l_k$ respectively. Standard order requires the levels of factor f_1 are in order $1, 2, \ldots, l_1$ and within each level of f_1 the levels of f_2 are in order $1, 2, \ldots, l_2$ and so on.

For an experiment with blocks the data is for block 1 then for block 2 etc. Within each block the data must be arranged so that the levels of factor f_1 are in order $1, 2, \ldots, l_1$ and within each level of f_1 the levels of f_2 are in order $1, 2, \ldots, l_2$ and so on. Any within block replication of treatment combinations must occur within the levels of f_k.

The ANOVA table is given in the following order. For a complete factorial experiment the first row is for blocks, if present, then the main effects of the factors in their order, e.g. f_1 followed by f_2 etc. These are then followed by all the two factor interactions then all the three factor interactions etc. The last two rows being for the residual and total sums of squares. The interactions are arranged in lexical order for the given order. For example, for the three factor interactions for a five factor experiment the 10 interactions would be in the following order:

$$\begin{array}{c} f_1f_2f_3 \\ f_1f_2f_4 \\ f_1f_2f_5 \\ f_1f_3f_4 \\ f_1f_3f_5 \\ f_1f_4f_5 \\ f_2f_3f_4 \\ f_2f_3f_5 \\ f_2f_4f_5 \\ f_3f_4f_5 \end{array}$$

4 References

[1] Cochran W G and Cox G M (1957) *Experimental Designs* Wiley

[2] Davis O L (1978) *The Design and Analysis of Industrial Experiments* Longman

[3] John J A and Quenouille M H (1977) *Experiments: Design and Analysis* Griffin

5 Parameters

1: N — INTEGER *Input*

On entry: the number of observations.

Constraints:

$N \geq 4$.

N must be a multiple of NBLOCK if NBLOCK > 1.

N must be a multiple of the number of treatment combinations, that is a multiple of $\prod_{i=1}^{k} \mathrm{LFAC}(i)$.

2: Y(N) — ***real*** array *Input*

On entry: the observations in standard order, see Section 3.

3: NFAC — INTEGER *Input*

On entry: the number of factors, k.

Constraint: NFAC $\geq$ 1.

4: LFAC(NFAC) — INTEGER array *Input*

On entry: LFAC(i) must contain the number of levels for the ith factor, $i = 1, 2, \ldots, k$.

Constraint: LFAC(i) ≥ 2 for $i = 1, 2, \ldots, k$.

5: NBLOCK — INTEGER *Input*

On entry: the number of blocks. If there are no blocks, set NBLOCK = 0 or 1.

Constraints:

NBLOCK $\geq$ 0.
If NBLOCK $\geq$ 2, N/NBLOCK must be a multiple of the number of treatment combinations, that is a multiple of $\prod_{i=1}^{k}$ LFAC(i).

6: INTER — INTEGER *Input*

On entry: the maximum number of factors in an interaction term. If no interaction terms are to be computed, set INTER = 0 or 1.

Constraint: 0 $\leq$ INTER $\leq$ NFAC.

7: IRDF — INTEGER *Input*

On entry: the adjustment to the residual and total degrees of freedom. The total degrees of freedom are set to N $-$ IRDF and the residual degrees of freedom adjusted accordingly. For examples of the use of IRDF see Section 8.

Constraint: IRDF $\geq$ 0.

8: MTERM — INTEGER *Input*

On entry: the maximum number of terms in the analysis of variance table, see Section 8.

Constraint: MTERM must be large enough to contain the terms specified by NFAC, INTER and NBLOCK. If the routine exits with IFAIL $\geq$ 2, the required minimum value of MTERM is returned in ITOTAL.

9: TABLE(MTERM,5) — ***real*** array *Output*

On exit: the first ITOTAL rows of TABLE contain the analysis of variance table. The first column contains the degrees of freedom, the second column contains the sum of squares, the third column (except for the row corresponding to the total sum of squares) contains the mean squares, i.e., the sums of squares divided by the degrees of freedom, and the fourth and fifth columns contain the F ratio and significance level, respectively (except for rows corresponding to the total sum of squares, and the residual sum of squares). All other cells of the table are set to zero.

The first row corresponds to the blocks and is set to zero if there are no blocks. The ITOTALth row corresponds to the total sum of squares for Y and the (TOTAL$-$1)th row corresponds to the residual sum of squares. The central rows of the table correspond to the main effects followed by the interaction if specified by INTER. The main effects are in the order specified by LFAC and the interactions are in lexical order, see Section 3.

10: ITOTAL — INTEGER *Output*

On exit: the row in TABLE corresponding to the total sum of squares. The number of treatment effects is ITOTAL $-$ 3.

11: TMEAN(MAXT) — ***real*** array *Output*

On exit: the treatment means. The position of the means for an effect is given by the index in IMEAN. For a given effect the means are in standard order, see Section 3.

12: MAXT — INTEGER *Input*

On entry: the maximum number of treatment means to be computed, see Section 8. If the value of MAXT is too small for the required analysis then the minimum number is returned in IMEAN(1).

Constraint: MAXT must be large enough for the number of means specified by LFAC and INTER; if INTER = NFAC then MAXT $\geq \prod_{i=1}^{k}(\mathrm{LFAC}(i)+1)-1$.

13: E(MAXT) — ***real*** array *Output*

On exit: the estimated effects in the same order as for the means in TMEAN.

14: IMEAN(MTERM) — INTEGER array *Output*

On exit: indicates the position of the effect means in TMEAN. The effect means corresponding to the first treatment effect in the ANOVA table are stored in TMEAN(1) up to TMEAN(IMEAN(1)). Other effect means corresponding to the ith treatment effect, $i = 1,2,\ldots,$ITOTAL $-$ 3, are stored in TMEAN(IMEAN($i-1$) + 1) up to TMEAN(IMEAN(i)).

15: SEMEAN(MTERM) — ***real*** array *Output*

On exit: the standard error of the difference between means corresponding to the ith treatment effect in the ANOVA table.

16: BMEAN(NBLOCK+1) — ***real*** array *Output*

On exit: BMEAN(1) contains the grand mean, if NBLOCK > 1, BMEAN(2) up to BMEAN(NBLOCK+1) contain the block means.

17: R(N) — ***real*** array *Output*

On exit: the residuals.

18: IWK(N+3∗NFAC) — INTEGER array *Workspace*

19: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, N < 4,
or NFAC < 1,
or NBLOCK < 0,
or INTER < 0,
or INTER > NFAC,
or IRDF < 0.

IFAIL = 2

On entry, LFAC(i) ≤ 1, for some $i = 1, 2, \ldots,$ NFAC,
or the value of MAXT is too small,
or the value of MTERM is too small,
or NBLOCK > 1 and N is not a multiple of NBLOCK,
or the number of plots per block is not a multiple of the number of treatment combinations.

IFAIL = 3

On entry, the values of Y are constant.

IFAIL = 4

There are no degrees of freedom for the residual or the residual sum of squares is zero. In either case the standard errors and F-statistics cannot be computed.

7 Accuracy

The block and treatment sums of squares are computed from the block and treatment residual totals. The residuals are updated as each effect is computed and the residual sum of squares computed directly from the residuals. This avoids any loss of accuracy in subtracting sums of squares.

8 Further Comments

The number of rows in the ANOVA table and the number of treatment means are given by the following formulae.

Let there be k factors with levels l_i for $i = 1, 2, \ldots, k$. and let t be the maximum number of terms in an interaction then the number of rows in the ANOVA tables is:

$$\sum_{i=1}^{t} \binom{k}{i} + 3.$$

The number of treatment means is:

$$\sum_{i=1}^{t} \prod_{j \in S_i} l_j,$$

where S_i is the set of all combinations of the k factors i at a time.

To estimate missing values the Healy and Westmacott procedure or its derivatives may be used, see [3]. This is an iterative procedure in which estimates of the missing values are adjusted by subtracting the corresponding values of the residuals. The new estimates are then used in the analysis of variance. This process is repeated until convergence. A suitable initial value may be the grand mean. When using this procedure IRDF should be set to the number of missing values plus one to obtain the correct degrees of freedom for the residual sum of squares.

For analysis of covariance the residuals are obtained from an analysis of variance of both the response variable and the covariates. The residuals from the response variable are then regressed on the residuals from the covariates using, say, G02CBF or G02DAF. The coefficients obtained from the regression can be examined for significance and used to produce an adjusted dependent variable using the original response variable and covariate. An approximate adjusted analysis of variance table can then be produced by using the adjusted dependent variable. In this case IRDF should be set to one plus the number of fitted covariates.

For designs such as Latin squares one more of the blocking factors has to be removed in a preliminary analysis before the final analysis. This preliminary analysis can be performed using G04BBF or a prior call to G04CAF if the data is reordered between calls. The residuals from the preliminary analysis are then input to G04CAF. In these cases IRDF should be set to the difference between N and the residual degrees of freedom from preliminary analysis. Care should be taken when using this approach as there is no check on the orthogonality of the two analyses.

9 Example

The data, given by John and Quenouille [3], is for the yield of turnips for a factorial experiment with two factors, the amount of phosphate with 6 levels and the amount of liming with 3 levels. The design was replicated in 3 blocks. The data is input and the analysis of variance computed. The analysis of variance table and tables of means with their standard errors are printed.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G04CAF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MAXF, MAXT, MTERM, BMAX
      PARAMETER        (NMAX=54,MAXF=2,MAXT=27,MTERM=6,BMAX=4)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INTER, IRDF, ITOTAL, J, K, L, N,
     +                 NBLOCK, NFAC, NTREAT
*     .. Local Arrays ..
      real             BMEAN(BMAX), E(MAXT), R(NMAX), SEMEAN(MTERM),
     +                 TABLE(MTERM,5), TMEAN(MAXT), Y(NMAX)
      INTEGER          IMEAN(MTERM), IWK(NMAX+3*MAXF), LFAC(MAXF)
*     .. External Subroutines ..
      EXTERNAL         G04CAF
*     .. Executable Statements ..
      WRITE (NOUT,FMT=*) 'G04CAF Example Program Results'
*     Skip heading in data file
      READ (NIN,FMT=*)
      READ (NIN,FMT=*) N, NBLOCK, NFAC, INTER
      IF (N.LE.NMAX .AND. NBLOCK.LE.BMAX-1 .AND. NFAC.LE.MAXF) THEN
         READ (NIN,FMT=*) (LFAC(J),J=1,NFAC)
         READ (NIN,FMT=*) (Y(I),I=1,N)
         IRDF = 0
         IFAIL = -1
*
         CALL G04CAF(N,Y,NFAC,LFAC,NBLOCK,INTER,IRDF,MTERM,TABLE,ITOTAL,
     +               TMEAN,MAXT,E,IMEAN,SEMEAN,BMEAN,R,IWK,IFAIL)
*
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*) ' ANOVA table'
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*)
     +     ' Source      df         SS           MS           F',
     +     '          Prob'
         WRITE (NOUT,FMT=*)
         K = 0
         IF (NBLOCK.GT.1) THEN
            K = K + 1
            WRITE (NOUT,FMT=99998) ' Blocks    ', (TABLE(1,J),J=1,5)
         END IF
         NTREAT = ITOTAL - 2 - K
         DO 20 I = 1, NTREAT
            WRITE (NOUT,FMT=99997) ' Effect  ', I, (TABLE(K+I,J),J=1,5)
   20    CONTINUE
```

```
         WRITE (NOUT,FMT=99998) ' Residual  ', (TABLE(ITOTAL-1,J),J=1,3)
         WRITE (NOUT,FMT=99998) ' Total     ', (TABLE(ITOTAL,J),J=1,2)
         WRITE (NOUT,FMT=*)
         WRITE (NOUT,FMT=*) ' Treatment Means and Standard Errors'
         WRITE (NOUT,FMT=*)
         K = 1
         DO 40 I = 1, NTREAT
            L = IMEAN(I)
            WRITE (NOUT,FMT=99996) ' Effect ', I
            WRITE (NOUT,FMT=*)
            WRITE (NOUT,FMT=99999) (TMEAN(J),J=K,L)
            WRITE (NOUT,FMT=*)
            WRITE (NOUT,FMT=99995) ' SE of difference in means  = ',
     +        SEMEAN(I)
            WRITE (NOUT,FMT=*)
            K = L + 1
   40    CONTINUE
      END IF
      STOP
*
99999 FORMAT (8F10.2)
99998 FORMAT (A,3X,F3.0,2X,2(F10.0,2X),F10.3,2X,F9.4)
99997 FORMAT (A,I2,3X,F3.0,2X,2(F10.0,2X),F10.3,2X,F9.4)
99996 FORMAT (A,I2)
99995 FORMAT (A,F10.2)
      END
```

9.2 Program Data

```
G04CAF Example Program Data
54 3 2 2   : N NBLOCK NFAC INTER
6 3        : LFAC

274 361 253 325 317 339 326 402 336 379 345 361 352 334 318 339 393 358
350 340 203 397 356 298 382 376 355 418 387 379 432 339 293 322 417 342
 82 297 133 306 352 361 220 333 270 388 379 274 336 307 266 389 333 353
```

9.3 Program Results

```
 G04CAF Example Program Results

  ANOVA table

   Source      df        SS          MS          F        Prob

 Blocks        2.      30119.      15059.      7.685     0.0018
 Effect   1    5.      73008.      14602.      7.451     0.0001
 Effect   2    2.      21596.      10798.      5.510     0.0085
 Effect   3   10.      31192.       3119.      1.592     0.1513
 Residual     34.      66628.       1960.
 Total        53.     222543.

  Treatment Means and Standard Errors

 Effect  1

    254.78    339.00    333.33    367.78    330.78    360.67
```

```
SE of difference in means  =      20.87

Effect  2

   334.28    353.78    305.11

SE of difference in means  =      14.76

Effect  3

   235.33    332.67    196.33    342.67    341.67    332.67    309.33    370.33
   320.33    395.00    370.33    338.00    373.33    326.67    292.33    350.00
   381.00    351.00

SE of difference in means  =      36.14
```

G04DAF – NAG Fortran Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G04DAF computes sum of squares for a user defined contrast between means.

2 Specification

```
      SUBROUTINE G04DAF(NT, TMEAN, IREP, RMS, RDF, NC, CT, LDCT, EST,
     1                  TABLE, LDT, TOL, USETX, TX, IFAIL)
      INTEGER           NT, IREP(NT), NC, LDCT, LDT, IFAIL
      real              TMEAN(NT), RMS, RDF, CT(LDCT,NC), EST(NC),
     1                  TABLE(LDT,5), TOL, TX(NT)
      LOGICAL           USETX
```

3 Description

In the analysis of designed experiments the first stage is to compute the basic analysis of variance table, the estimate of the error variance (the residual or error mean square), $\hat{\sigma}^2$, and the (variance ratio) F-statistic for the t treatments. If this F test is significant then the second stage of the analysis is to explore which treatments are significantly different.

If there is a structure to the treatments then this may lead to hypotheses that can be defined before the analysis and tested using linear contrasts. For example, if the treatments were three different fixed temperatures, say 18, 20 and 22, and an uncontrolled temperature (denoted by N) then the following contrasts might be of interest.

	18	20	22	N
(a)	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	-1
(b)	-1	0	1	0

The first represents the average difference between the controlled temperatures and the uncontrolled temperature. The second represents the linear effect of an increasing fixed temperature.

For a randomised complete block design or a completely randomised design, let the treatment means be $\hat{\tau}_i$, $i = 1,2,\ldots,t$, and let the jth contrast be defined by λ_{ij}, $i = 1,2,\ldots,t$, then the estimate of the contrast is simply:

$$\Lambda_j = \sum_{i=1}^{t} \hat{\tau}_i \lambda_{ij}$$

and the sum of squares for the contrast is:

$$SS_j = \frac{\Lambda_j^2}{\sum_{i=1}^{t} \lambda_{ij}^2 / n_i} \tag{1}$$

where n_i is the number of observations for the ith treatment. Such a contrast has one degree of freedom so that the appropriate F-statistic is $SS_j/\hat{\sigma}^2$.

The two contrasts λ_{ij} and $\lambda_{ij'}$ are orthogonal if $\sum_{i=1}^{t} \lambda_{ij}\lambda_{ij'} = 0$ and the contrast λ_{ij} is orthogonal to the overall mean if $\sum_{i=1}^{t} \lambda_{ij} = 0$. In practice these sums will be tested against a small quantity, ϵ. If each of a set of contrasts is orthogonal to the mean and they are all mutually orthogonal then the contrasts provide a partition of the treatment sum of squares into independent components. Hence the resulting F-tests are independent.

If the treatments come from a design in which treatments are not orthogonal to blocks then the sum of squares for a contrast is given by:

$$SS_j = \frac{\Lambda_j \Lambda_j^*}{\sum_{i=1}^t \lambda_{ij}^2/n_i} \tag{2}$$

where

$$\Lambda_j^* = \sum_{i=1}^t \tau_i^* \lambda_{ij}$$

with τ_i^*, $i = 1,2,\ldots,t$ being adjusted treatment means computed by first eliminating blocks then computing the treatment means from the block adjusted observations without taking into account the non-orthogonality between treatments and blocks. For further details see John [2] and Morgan [3].

4 References

[1] Cochran W G and Cox G M (1957) *Experimental Designs* Wiley

[2] John J A (1987) *Cyclic Designs* Chapman and Hall

[3] Morgan G W (1993) Analysis of variance using the NAG Fortran Library: Examples from Cochran and Cox *NAG Technical Report TR 3/93* NAG Ltd, Oxford

[4] Winer B J (1970) *Statistical Principles in Experimental Design* McGraw-Hill

5 Parameters

1: NT — INTEGER *Input*

On entry: the number of treatment means, t.

Constraint: NT ≥ 2.

2: TMEAN(NT) — ***real*** array *Input*

On entry: the treatment means, $\hat{\tau}_i$, $i = 1,2,\ldots,t$.

3: IREP(NT) — INTEGER array *Input*

On entry: the replication for each treatment mean, n_i, $i = 1,2,\ldots,t$.

4: RMS — ***real*** *Input*

On entry: the residual mean square, $\hat{\sigma}^2$.

Constraint: RMS > 0.0.

5: RDF — ***real*** *Input*

On entry: the residual degrees of freedom.

Constraint: RDF ≥ 1.0.

6: NC — INTEGER *Input*

On entry: the number of contrasts.

Constraint: NC ≥ 1.

7: CT(LDCT,NC) — ***real*** array *Input*

On entry: the columns of CT must contain the NC contrasts, that is CT(i,j) must contain λ_{ij} for $i = 1,2,\ldots,t$; $j = 1,2,\ldots,$NC.

8: LDCT — INTEGER *Input*

On entry: the first dimension of the array CT as declared in the (sub)program from which G04DAF is called.

Constraint: LDCT $\geq$ NT.

9: EST(NC) — ***real*** array *Output*

On exit: the estimates of the contrast, Λ_j, $j = 1,2,\ldots,\mathrm{NC}$.

10: TABLE(LDT,5) — ***real*** array *Output*

On exit: the rows of the analysis of variance table for the contrasts. For each row column 1 contains the degrees of freedom, column 2 contains the sum of squares, column 3 contains the mean square, column 4 the F-statistic and column 5 the significance level for the contrast. Note that the degrees of freedom are always one and so the mean square equals the sum of squares.

11: LDT — INTEGER *Input*

On entry: the first dimension of the array TABLE as declared in the (sub)program from which G04DAF is called.

Constraint: LDT $\geq$ NC.

12: TOL — ***real*** *Input*

On entry: the tolerance, ϵ used to check if the contrasts are orthogonal and if they are orthogonal to the mean. If TOL ≤ 0.0 the value ***machine precision*** is used.

13: USETX — LOGICAL *Input*

On entry: if USETX = .TRUE. the means τ_i^* are provided in TX and the formula (2) is used instead of formula (1). If USETX = .FALSE. formula (1) is used and TX is not referenced.

14: TX(NT) — ***real*** array *Input*

On entry: if USETX = .TRUE. TX must contain the means τ_i^*, $i = 1,2,\ldots,t$.

15: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL $\neq 0$ on exit, users are recommended to set IFAIL to -1 before entry. **It is then essential to test the value of IFAIL on exit.**

6 Errors and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings specified by the routine:

IFAIL = 1

On entry, NC < 1,
or NT < 2,
or LDCT $<$ NT,
or LDT $<$ NC,
or RMS ≤ 0.0,
or RDF < 1.0.

IFAIL = 2

On entry, a contrast is not orthogonal to the mean,

or at least two contrasts are not orthogonal.

If IFAIL = 2 full results are returned but they should be interpreted with care.

7 Accuracy

The computations are stable.

8 Further Comments

If the treatments have a factorial structure G04CAF should be used and if the treatments have no structure the means can be compared using G04DBF.

9 Example

The data is given in Cochran and Cox [3] and is for a completely randomised experiment on potato scab with seven treatments representing amounts of sulphur applied, whether the application was in spring or autumn and a control treatment. The one-way anova is computed using G02BBF. Two contrasts are analysed, one comparing the control with use of sulphur, the other comparing spring with autumn application.

9.1 Example Text

Note: the listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G04DAF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NTMAX, NBMAX
      PARAMETER        (NMAX=32,NTMAX=7,NBMAX=1)
*     .. Local Scalars ..
      real             GMEAN, RDF, RMS, TOL
      INTEGER          I, IFAIL, IRDF, J, LDT, N, NBLOCK, NC, NT
*     .. Local Arrays ..
      real             BMEAN(NBMAX), C(NTMAX,NTMAX), CT(NTMAX,NTMAX),
     +                 EF(NTMAX), EST(NTMAX), R(NMAX), TABLE(NTMAX+4,5),
     +                 TMEAN(NTMAX), TX(NTMAX), WK(NTMAX*NTMAX+NTMAX),
     +                 Y(NMAX)
      INTEGER          IREP(NTMAX), IT(NMAX)
      CHARACTER*11     NAMES(NTMAX)
*     .. External Subroutines ..
      EXTERNAL         G04BBF, G04DAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G04DAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NT
      IF (N.LE.NMAX .AND. NT.LE.NTMAX) THEN
         READ (NIN,*) (Y(I),I=1,N)
         READ (NIN,*) (IT(I),I=1,N)
         TOL = 0.000005e0
         IRDF = 0
         NBLOCK = 1
         LDT = NTMAX + 4
         IFAIL = -1
         CALL G04BBF(N,Y,NBLOCK,NT,IT,GMEAN,BMEAN,TMEAN,TABLE,LDT,C,
     +               NTMAX,IREP,R,EF,TOL,IRDF,WK,IFAIL)
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' ANOVA table'
         WRITE (NOUT,*)
         WRITE (NOUT,*)
     +     ' Source        df         SS          MS          F',
     +     '         Prob'
         WRITE (NOUT,*)
         WRITE (NOUT,99998) ' Treatments', (TABLE(2,J),J=1,5)
         WRITE (NOUT,99998) ' Residual  ', (TABLE(3,J),J=1,3)
         WRITE (NOUT,99998) ' Total     ', (TABLE(4,J),J=1,2)
*
         RMS = TABLE(3,3)
         RDF = TABLE(3,1)
         READ (NIN,*) NC
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Orthogonal Contrasts'
         WRITE (NOUT,*)
         DO 20 I = 1, NC
            READ (NIN,*) (CT(J,I),J=1,NT)
            READ (NIN,99999) NAMES(I)
   20    CONTINUE
         CALL G04DAF(NT,TMEAN,IREP,RMS,RDF,NC,CT,NTMAX,EST,TABLE(5,1),
     +               LDT,TOL,.FALSE.,TX,IFAIL)
         DO 40 I = 1, NC
            WRITE (NOUT,99998) NAMES(I), (TABLE(I+4,J),J=1,5)
   40    CONTINUE
      END IF
      STOP
*
99999 FORMAT (A)
99998 FORMAT (A,3X,F3.0,2X,2(F10.1,2X),F10.3,2X,F9.4)
      END
```

9.2 Example Data

```
G04DAF Example Program Data

32 7

12 10 24 29 30 18 32 26
9 9 16 4 30 7 21 9 16 10 18 18
18 24 12 19 10 4 4 5 17 7 16 17

1 1 1 1 1 1 1 1
2 2 2 2 3 3 3 3 4 4 4 4
5 5 5 5 6 6 6 6 7 7 7 7

2

6 -1 -1 -1 -1 -1 -1
 Cntl v S
0 1 -1 1 -1 1 -1
```

```
Spring v A
```

9.3 Example Results

```
G04DAF Example Program Results

 ANOVA table

  Source        df         SS           MS          F          Prob

Treatments     6.        972.3        162.1       3.608       0.0103
Residual      25.       1122.9         44.9
Total         31.       2095.2

 Orthogonal Contrasts

Cntl v S       1.        518.0        518.0      11.533       0.0023
Spring v A     1.        228.2        228.2       5.080       0.0332
```

G04DBF – NAG Fortran Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G04DBF computes simultaneous confidence intervals for the differences between means. It is intended for use after G04BBF or G04BCF.

2 Specification

```
      SUBROUTINE G04DBF(TYPE, NT, TMEAN, RDF, C, LDC, CLEVEL, CIL, CIU,
     1                  ISIG, IFAIL)
      INTEGER           NT, LDC, ISIG(NT*(NT-1)/2), IFAIL
      real              TMEAN(NT), RDF, C(LDC,NT), CLEVEL,
     1                  CIL(NT*(NT-1)/2), CIU(NT*(NT-1)/2)
      CHARACTER*1       TYPE
```

3 Description

In the computation of analysis of a designed experiment the first stage is to compute the basic analysis of variance table, the estimate of the error variance (the residual or error mean square), $\hat{\sigma}^2$, the residual degress of freedom, ν, and the (variance ratio) F-statistic for the t treatments. The second stage of the analysis is to compare the treatment means. If the treatments have no structure, for example the treatments are different varieties, rather than being structured, for example a set of different temperatures, then a multiple comparison procedure can be used.

A multiple comparison procedure looks at all possible pairs of means and either computes confidence intervals for the difference in means or performs a suitable test on the difference. If there are t treatments then there are $t(t-1)/2$ comparisons to be considered. In tests the type 1 error or significance level is the probability that the result is considered to be significant when there is no difference in the means. If the usual t-test is used with, say, a five percent significance level then the type 1 error for all $k = t(t-1)/2$ tests will be much higher. If the tests were independent then if each test is carried out at the 100α percent level then the overall type 1 error would be $\alpha^* = 1-(1-\alpha)^k \simeq k\alpha$. In order to provide an overall protection the individual tests, or confidence intervals, would have to be carried out at a value of α such that α^* is the required significance level, e.g. five percent.

The $100(1-\alpha)$ percent confidence interval for the difference in two treatment means, $\hat{\tau}_i$ and $\hat{\tau}_j$ is given by

$$(\hat{\tau}_i - \hat{\tau}_j) \pm T^*_{(\alpha,\nu,t)} se(\hat{\tau}_i - \hat{\tau}_j),$$

where $se()$ denotes the standard error of the difference in means and $T^*_{(\alpha,\nu,t)}$ is an appropriate percentage point from a distribution. There are several possible choices for $T^*_{(\alpha,\nu,t)}$. These are:

(a) $\frac{1}{2}q_{(1-\alpha,\nu,t)}$, the studentised range statistic, see G01FMF. It is the appropriate statistic to compare the largest mean with the smallest mean. This is known as Tukey–Kramer method.

(b) $t_{(\alpha/k,\nu)}$, this is the Bonferroni method.

(c) $t_{(\alpha_0,\nu)}$, where $\alpha_0 = 1-(1-\alpha)^{1/k}$, this is known as the Dunn–Sidak method.

(d) $t_{(\alpha,\nu)}$, this is known as Fisher's LSD (least significant difference) method. It should only be used if the overall F-test is significant, the number of treatment comparisons is small and were planned before the analysis.

(e) $\sqrt{(k-1)F_{1-\alpha,k-1,\nu}}$ where $F_{1-\alpha,k-1,\nu}$ is the deviate corresponding to a lower tail probability of $1-\alpha$ from an F-distribution with $k-1$ and ν degrees of freedom. This is Scheffe's method.

In cases (b), (c) and (d), $t_{(\alpha,\nu)}$ denotes the α two-tail significance level for the Student's t-distribution with ν degrees of freedom, see G01FBF.

The Scheffe method is the most conservative, followed closely by the Dunn–Sidak and Tukey–Kramer methods.

To compute a test for the difference between two means the statistic,

$$\frac{\hat{\tau}_i - \hat{\tau}_j}{se(\hat{\tau}_i - \hat{\tau}_j)}$$

is compared with the appropriate value of $T^*_{(\alpha,\nu,t)}$.

4 References

[1] Kotz S and Johnson N L (ed.) (1985) Multiple range and associated test procedures *Encyclopedia of Statistical Sciences* **5** Wiley, New York

[2] Kotz S and Johnson N L (ed.) (1985) Multiple comparison *Encyclopedia of Statistical Sciences* **5** Wiley, New York

[3] Winer B J (1970) *Statistical Principles in Experimental Design* McGraw-Hill

5 Parameters

1: TYPE — CHARACTER*1 *Input*

On entry: indicates which method is to be used.

If TYPE = 'T', the Tukey–Kramer method is used.
If TYPE = 'B', the Bonferroni method is used.
If TYPE = 'D', the Dunn–Sidak method is used.
If TYPE = 'L', the Fisher LSD method is used.
If TYPE = 'S', the Scheffe's method is used.

Constraint: TYPE = 'T','B','D', 'L' or 'S'.

2: NT — INTEGER *Input*

On entry: the number of treatment means, t.

Constraint: NT $\geq$ 2.

3: TMEAN(NT) — ***real*** array *Input*

On entry: the treatment means, $\hat{\tau}_i$, i = 1,2,...,t.

4: RDF — ***real*** *Input*

On entry: the residual degrees of freedom, ν.

Constraint: RDF $\geq$ 1.0.

5: C(LDC,NT) — ***real*** array *Input*

On entry: the strictly lower triangular part of C must contain the standard errors of the differences between the means as returned by G04BBF and G04BCF. That is C(i,j), $i > j$, contains the standard error of the difference between the ith and jth mean in TMEAN.

Constraint: $C(i,j) > 0.0$, $i = 2,3,\ldots,t$; $j = 1,2,\ldots,i-1$.

6: LDC — INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which G04DBF is called.

Constraint: LDC $\geq$ NT.

7: CLEVEL — ***real*** *Input*

On entry: the required confidence level for the computed intervals, $(1-\alpha)$.

Constraint: $0.0 <$ CLEVEL < 1.0.

8: CIL(NT*(NT−1)/2) — ***real*** array *Output*

On exit: the $((i-1)(i-2)/2+j)$th element contains the lower limit to the confidence interval for the difference between ith and jth means in TMEAN, $i=2,3,\ldots,t$; $j=1,2,\ldots,i-1$.

9: CIU(NT*(NT−1)/2) — ***real*** array *Output*

On exit: the $((i-1)(i-2)/2+j)$th element contains the upper limit to the confidence interval for the difference between ith and jth means in TMEAN, $i=2,3,\ldots,t$; $j=1,2,\ldots,i-1$.

10: ISIG(NT*(NT−1)/2) — INTEGER array *Output*

On exit: the $((i-1)(i-2)/2+j)$th element indicates if the difference between ith and jth means in TMEAN is significant, $i=2,3,\ldots,t$; $j=1,2,\ldots,i-1$. If the difference is significant then the returned value is 1; otherwise the returned value is 0.

11: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings specified by the routine:

IFAIL = 1

On entry, NT < 2,
or LDC $<$ NT,
or RDF < 1.0,
or CLEVEL ≤ 0.0,
or CLEVEL ≥ 1.0,
or TYPE $\neq$ 'T', 'B', 'D', 'L' or 'S'.

IFAIL = 2

On entry, $C(i,j) \leq 0.0$ for some i,j, $i=2,3,\ldots,t$; $j=1,2,\ldots,i-1$.

IFAIL = 3

There has been a failure in the computation of the studentized range statistic. This is an unlikely error. Try using a small value of CLEVEL.

7 Accuracy

For the accuracy of the percentage point statistics see G01FMF and G01FBF.

8 Further Comments

If the treatments have a structure then the use of linear contrasts as computed by G04DAF may be more appropriate.

An alternative approach to one used in this routine is the sequential testing of the Student-Newman–Keuls procedure. This, in effect, uses the Tukey–Kramer method but first ordering the treatment means and examining only subsets of the treatment means in which the largest and smallest are significantly different. At each stage the third parameter of the Studentised range statistic is the number of means in the subset rather than the total number of means.

9 Example

In the example taken from Winer [3] a completely randomised design with unequal treatment replication is analysed using G04BBF and then confidence intervals are computed by G04DBF using the Tukey–Kramer method.

9.1 Example Text

Note: the listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G04DBF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NTMAX, NBMAX
      PARAMETER        (NMAX=26,NTMAX=4,NBMAX=1)
*     .. Local Scalars ..
      real             CLEVEL, GMEAN, RDF, TOL
      INTEGER          I, IFAIL, IJ, IRDF, J, N, NBLOCK, NT
      CHARACTER        TYPE
*     .. Local Arrays ..
      real             BMEAN(NBMAX), C(NTMAX,NTMAX),
     +                 CIL(NTMAX*(NTMAX-1)/2), CIU(NTMAX*(NTMAX-1)/2),
     +                 EF(NTMAX), R(NMAX), TABLE(4,5), TMEAN(NTMAX),
     +                 WK(NTMAX*NTMAX+NTMAX), Y(NMAX)
      INTEGER          IREP(NTMAX), ISIG(NTMAX*(NTMAX-1)/2), IT(NMAX)
      CHARACTER        STAR(2)
*     .. External Subroutines ..
      EXTERNAL         G04BBF, G04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G04DBF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NT
      IF (N.LE.NMAX .AND. NT.LE.NTMAX) THEN
         READ (NIN,*) (Y(I),I=1,N)
         READ (NIN,*) (IT(I),I=1,N)
         TOL = 0.000005e0
         IRDF = 0
         NBLOCK = 1
         IFAIL = -1
         CALL G04BBF(N,Y,NBLOCK,NT,IT,GMEAN,BMEAN,TMEAN,TABLE,4,C,NTMAX,
     +               IREP,R,EF,TOL,IRDF,WK,IFAIL)
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' ANOVA table'
```

```
            WRITE (NOUT,*)
            WRITE (NOUT,*)
     +        ' Source         df          SS            MS            F',
     +        '           Prob'
            WRITE (NOUT,*)
            WRITE (NOUT,99998) ' Treatments', (TABLE(2,J),J=1,5)
            WRITE (NOUT,99998) ' Residual  ', (TABLE(3,J),J=1,3)
            WRITE (NOUT,99998) ' Total     ', (TABLE(4,J),J=1,2)
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Treatment means'
            WRITE (NOUT,*)
            WRITE (NOUT,99999) (TMEAN(J),J=1,NT)
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Simultaneous Confidence Intervals'
            WRITE (NOUT,*)
            RDF = TABLE(3,1)
            READ (NIN,*) TYPE, CLEVEL
*
            CALL G04DBF(TYPE,NT,TMEAN,RDF,C,NTMAX,CLEVEL,CIL,CIU,ISIG,
     +                  IFAIL)
*
            STAR(2) = '*'
            STAR(1) = ' '
            IJ = 0
            DO 40 I = 1, NT
               DO 20 J = 1, I - 1
                  IJ = IJ + 1
                  WRITE (NOUT,99997) I, J, CIL(IJ), CIU(IJ),
     +              STAR(ISIG(IJ)+1)
   20          CONTINUE
   40       CONTINUE
         END IF
         STOP
*
99999 FORMAT (10F8.3)
99998 FORMAT (A,3X,F3.0,2X,2(F10.1,2X),F10.3,2X,F9.4)
99997 FORMAT (2X,2I2,3X,2(F10.3,3X),A)
      END
```

9.2 Example Data

```
G04DBF Example Program Data

26 4

 3  2  4  3  1  5
 7  8  4 10  6
 3  2  1  2  4  2  3  1
10 12  8  5 12 10  9

1 1 1 1 1 1
2 2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4

'T' .95
```

9.3 Example Results

```
G04DBF Example Program Results

 ANOVA table

  Source       df        SS          MS          F         Prob

Treatments    3.       239.9        80.0      24.029     0.0000
Residual     22.        73.2         3.3
Total        25.       313.1

 Treatment means

  3.000   7.000   2.250   9.429

 Simultaneous Confidence Intervals

  2 1         0.933         7.067   *
  3 1        -3.486         1.986
  3 2        -7.638        -1.862   *
  4 1         3.610         9.247   *
  4 2        -0.538         5.395
  4 3         4.557         9.800   *
```

G04EAF – NAG Fortran Library Routine Document

Note: Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

G04EAF computes orthogonal polynomial or dummy variables for a factor or classification variable.

2 Specification

```
SUBROUTINE G04EAF(TYPE, N, LEVELS, IFACT, X, LDX, V, REP, IFAIL)
INTEGER          N, LEVELS, IFACT(N), LDX, IFAIL
real             X(LDX,*), V(*), REP(LEVELS)
CHARACTER*1      TYPE
```

3 Description

In the analysis of an experimental design using a general linear model the factors or classification variables that specify the design have to be coded as dummy variables. G04EAF computes dummy variables that can then be used in the fitting of the general linear model using G02DAF.

If the factor of length n has k levels then the simplest representation is to define k dummy variables, X_j such that $X_j = 1$ if the factor is at level j and 0 otherwise for $j = 1,2,...,k$. However, there is usually a mean included in the model and the sum of the dummy variables will be aliased with the mean. To avoid the extra redundant parameter $k-1$ dummy variables can be defined as the contrasts between one level of the factor, the reference level and the remaining levels. If the reference level is the first level then the dummy variables can be defined as $X_j = 1$ if the factor is at level j and 0 otherwise for $j = 2,3,\ldots,k$. Alternatively, the last level can be used as the reference level.

A second way of defining the $k-1$ dummy variables is to use a Helmert matrix in which levels $2,3,\ldots,k$ are compared with the average effect of the previous levels. For example if $k = 4$ then the contrasts would be:

$$\begin{array}{cccc} 1 & -1 & -1 & -1 \\ 2 & 1 & -1 & -1 \\ 3 & 0 & 2 & -1 \\ 4 & 0 & 0 & 3 \end{array}$$

Thus variable j, for $j = 1,2,\ldots,k-1$ is given by

$X_j = -1$ if factor is at level less than $j+1$

$X_j = \sum_{i=1}^{j} r_i/r_{j+1}$ if factor is at level $j+1$

$X_j = 0$ if factor is at level greater than $j+1$

where r_j is the number of replicates of level j.

If the factor can be considered as a set of values from an underlying continuous variable then the factor can be represented by a set of $k-1$ orthogonal polynomials representing the linear, quadratic etc. effects of the underlying variable. The orthogonal polynomial is computed using Forsythe's algorithm [2], see Cooper [1]. The values of the underlying continuous variable represented by the factor levels have to be supplied to the routine.

The orthogonal polynomials are standardized so that the sum of squares for each dummy variable is one. For the other methods integer (± 1) representations are retained except that in the Helmert representation the code of level $j+1$ in dummy variable j will be a fraction.

4 References

[1] Cooper B E (1968) Algorithm AS 10. The use of orthogonal polynomials *Appl. Statist.* **17** 283–287

[2] Forsythe G E (1957) Generation and use of orthogonal polynomials for data fitting with a digital computer *J. Soc. Indust. Appl. Math.* **5** 74–88

5 Parameters

1: TYPE — CHARACTER*1 *Input*

On entry: the type of dummy variable to be computed.

If TYPE = 'P', an orthogonal Polynomial representation is computed.
If TYPE = 'H', a Helmert matrix representation is computed.
If TYPE = 'F', the contrasts relative to the First level are computed.
If TYPE = 'L', the contrasts relative to the Last level are computed.
If TYPE = 'C', a Complete set of dummy variables is computed.

Constraint: TYPE = 'P', 'H', 'F', 'L' or 'C'.

2: N — INTEGER *Input*

On entry: the number of observations for which the dummy variables are to be computed, n.

Constraint: N $\geq$ LEVELS.

3: LEVELS — INTEGER *Input*

On entry: the number of levels of the factor, k.

Constraint: LEVELS ≥ 2.

4: IFACT(N) — INTEGER array *Input*

On entry: the n values of the factor.

Constraint: $1 \leq$ IFACT(i) $\leq$ LEVELS, for $i = 1, 2, \ldots, n$.

5: X(LDX,*) — ***real*** array *Output*

Note: the second dimension of the array X must be at least LEVELS−1 if TYPE = 'P', 'H', 'F' or 'L' and LEVELS if TYPE = 'C'.

On exit: the n by k^* matrix of dummy variables, where $k^* = k - 1$ if TYPE = 'P', 'H', 'F' or 'L' and $k^* = k$ if TYPE = 'C'.

6: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G04EAF is called.

Constraint: LDX $\geq$ N.

7: V(*) — ***real*** array *Input*

Note: the dimension of the array V must be at least LEVELS if TYPE = 'P' and 1 otherwise.

On entry: if TYPE = 'P' the k distinct values of the underlying variable for which the orthogonal polynomial is to be computed. If TYPE $\neq$ 'P' V is not referenced.

Constraint: if TYPE = 'P' then the k values of V must be distinct.

8: REP(LEVELS) — ***real*** array *Output*

On exit: the number of replications for each level of the factor, r_i, $i = 1, 2, \ldots, k$.

9: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, LEVELS < 2,
or N < LEVELS,
or LDX < N,
or TYPE ≠ 'P', 'H', 'F', 'L' or 'C'.

IFAIL = 2

On entry, a value of IFACT is not in the range $1 \leq \text{IFACT}(i) \leq \text{LEVELS}$, for $i = 1,2,\ldots,n$,
or TYPE = 'P' and not all values of V are distinct,
or not all levels are represented in IFACT.

IFAIL= 3

An orthogonal polynomial has all values zero. This will be due to some values of V being very close together. Note this can only occur if TYPE = 'P'.

7 Accuracy

The computations are stable.

8 Further Comments

Other routines for fitting polynomials can be found in Chapter E02.

9 Example

Data are read in from an experiment with four treatments and three observations per treatment with the treatment coded as a factor. G04EAF is used to compute the required dummy variables and the model is then fitted by G02DAF.

9.1 Example Text

Note: the listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G04EAF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          MMAX, NMAX
      PARAMETER        (MMAX=5,NMAX=12)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             RSS, TOL
      INTEGER          I, IDF, IFAIL, IP, IRANK, J, LDX, LEVELS, M, N
      LOGICAL          SVD
      CHARACTER        MEAN, TYPE, WEIGHT
*     .. Local Arrays ..
      real             B(MMAX), COV((MMAX*MMAX+MMAX)/2), H(NMAX),
     +                  P(MMAX*(MMAX+2)), Q(NMAX,MMAX+1), REP(MMAX),
     +                  RES(NMAX), SE(MMAX), V(MMAX),
     +                  WK(MMAX*MMAX+5*(MMAX-1)), WT(NMAX), X(NMAX,MMAX),
     +                  Y(NMAX)
      INTEGER           IFACT(NMAX), ISX(MMAX)
*     .. External Subroutines ..
      EXTERNAL          G02DAF, G04EAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G04EAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, LEVELS, TYPE, WEIGHT, MEAN
      WRITE (NOUT,*)
      IF (N.LE.NMAX .AND. LEVELS.LE.MMAX) THEN
         IF (WEIGHT.EQ.'W' .OR. WEIGHT.EQ.'w') THEN
            DO 20 I = 1, N
               READ (NIN,*) IFACT(I), Y(I), WT(I)
   20       CONTINUE
         ELSE
            DO 40 I = 1, N
               READ (NIN,*) IFACT(I), Y(I)
   40       CONTINUE
         END IF
         IF (TYPE.EQ.'P' .OR. TYPE.EQ.'p') THEN
            READ (NIN,*) (V(J),J=1,LEVELS)
         END IF
*
*        Calculate dummy variables
*
         LDX = NMAX
         IFAIL = 0
*
         CALL G04EAF(TYPE,N,LEVELS,IFACT,X,LDX,V,REP,IFAIL)
*
         IF (TYPE.EQ.'C' .OR. TYPE.EQ.'c') THEN
            M = LEVELS
         ELSE
            M = LEVELS - 1
         END IF
         DO 60 J = 1, M
            ISX(J) = 1
   60    CONTINUE
         IP = M
         IF (MEAN.EQ.'M' .OR. MEAN.EQ.'m') IP = IP + 1
*        Set tolerance
         TOL = 0.00001e0
         IFAIL = 0
*
         CALL G02DAF(MEAN,WEIGHT,N,X,LDX,M,ISX,IP,Y,WT,RSS,IDF,B,SE,COV,
     +               RES,H,Q,NMAX,SVD,IRANK,P,TOL,WK,IFAIL)
*
         IF (SVD) THEN
            WRITE (NOUT,99999) 'Model not of full rank, rank = ', IRANK
            WRITE (NOUT,*)
         END IF
         WRITE (NOUT,99998) 'Residual sum of squares = ', RSS
         WRITE (NOUT,99999) 'Degrees of freedom = ', IDF
         WRITE (NOUT,*)
```

```
            WRITE (NOUT,*) 'Variable   Parameter estimate   Standard error'
            WRITE (NOUT,*)
            DO 80 J = 1, IP
               WRITE (NOUT,99997) J, B(J), SE(J)
   80       CONTINUE
         END IF
         STOP
*
99999 FORMAT (1X,A,I4)
99998 FORMAT (1X,A,e12.4)
99997 FORMAT (1X,I6,2e20.4)
      END
```

9.2 Example Data

```
G04EAF Example Program Data
 12 4 'C' 'U' 'M'
1 33.63
4 39.62
2 38.18
3 41.46
4 38.02
2 35.83
4 35.99
1 36.58
3 42.92
1 37.80
3 40.43
2 37.89
```

9.3 Example Results

```
 G04EAF Example Program Results

 Model not of full rank, rank =    4

 Residual sum of squares =   0.2223E+02
 Degrees of freedom =    8

 Variable   Parameter estimate   Standard error

      1          0.3056E+02          0.3849E+00
      2          0.5447E+01          0.8390E+00
      3          0.6743E+01          0.8390E+00
      4          0.1105E+02          0.8390E+00
      5          0.7320E+01          0.8390E+00
```

Chapter G05 – Random Number Generators

Note. Please refer to the Users' Note for your implementation to check that a routine is available.

Routine Name	Mark of Introduction	Purpose
G05CAF	6	Pseudo-random real numbers, uniform distribution over (0,1)
G05CBF	6	Initialise random number generating routines to give repeatable sequence
G05CCF	6	Initialise random number generating routines to give non-repeatable sequence
G05CFF	6	Save state of random number generating routines
G05CGF	6	Restore state of random number generating routines
G05DAF	6	Pseudo-random real numbers, uniform distribution over (a,b)
G05DBF	6	Pseudo-random real numbers, (negative) exponential distribution
G05DCF	6	Pseudo-random real numbers, logistic distribution
G05DDF	6	Pseudo-random real numbers, Normal distribution
G05DEF	6	Pseudo-random real numbers, lognormal distribution
G05DFF	6	Pseudo-random real numbers, Cauchy distribution
G05DHF	6	Pseudo-random real numbers, χ^2 distribution
G05DJF	6	Pseudo-random real numbers, Student's t-distribution
G05DKF	6	Pseudo-random real numbers, F-distribution
G05DPF	8	Pseudo-random real numbers, Weibull distribution
G05DRF	15	Pseudo-random integer, Poisson distribution
G05DYF	6	Pseudo-random integer from uniform distribution
G05DZF	6	Pseudo-random logical (boolean) value
G05EAF	10	Set up reference vector for multivariate Normal distribution
G05EBF	6	Set up reference vector for generating pseudo-random integers, uniform distribution
G05ECF	6	Set up reference vector for generating pseudo-random integers, Poisson distribution
G05EDF	6	Set up reference vector for generating pseudo-random integers, binomial distribution
G05EEF	6	Set up reference vector for generating pseudo-random integers, negative binomial distribution
G05EFF	6	Set up reference vector for generating pseudo-random integers, hypergeometric distribution
G05EGF	8	Set up reference vector for univariate ARMA time series model
G05EHF	10	Pseudo-random permutation of an integer vector
G05EJF	10	Pseudo-random sample from an integer vector
G05EWF	8	Generate next term from reference vector for ARMA time series model
G05EXF	6	Set up reference vector from supplied cumulative distribution function or probability distribution function
G05EYF	6	Pseudo-random integer from reference vector
G05EZF	10	Pseudo-random multivariate Normal vector from reference vector
G05FAF	14	Generates a vector of random numbers from a uniform distribution
G05FBF	14	Generates a vector of random numbers from an (negative) exponential distribution
G05FDF	14	Generates a vector of random numbers from a Normal distribution
G05FEF	15	Generates a vector of pseudo-random numbers from a beta distribution
G05FFF	15	Generates a vector of pseudo-random numbers from a gamma distribution
G05FSF	16	Generates vector of pseudo-random variates from Von Mises distribution
G05GAF	16	Computes random orthogonal matrix
G05GBF	16	Computes random correlation matrix

G05HDF	15	Generates a realisation of a multivariate time series from a VARMA model

Chapter G05

Random Number Generators

Contents

1 Scope of the Chapter

This chapter is concerned with the generation of sequences of independent pseudo-random numbers from various distributions, and the generation of pseudo-random time series from specified time-series models.

2 Background to the Problems

A sequence of pseudo-random numbers is a sequence of numbers generated in some systematic way such that its statistical properties are as close as possible to those of true random numbers: for example, negligible correlation between consecutive numbers. The most common method used is a **multiplicative congruential** algorithm defined as:

$$n_i = (a \times n_{i-1}) \bmod m \tag{1}$$

The integers n_i are then divided by m to give uniformly distributed random numbers lying in the interval (0,1).

The NAG generator uses the values $a = 13^{13}$ and $m = 2^{59}$; for further details see G05CAF. This generator gives a **cycle length** (i.e., the number of random numbers before the sequence starts repeating itself) of 2^{57}. A good rule of thumb is never to use more numbers than the square root of the cycle length in any one experiment as the statistical properties are impaired. For closely related reasons, breaking numbers down into their bit patterns and using individual bits may cause trouble.

The sequence given in (1) needs an initial value n_0, known as the **seed**. The use of the same seed will lead to the same sequence of numbers. One method of obtaining the seed is to use the real-time clock; this will give a non-repeatable sequence. It is important to note that the statistical properties of the random numbers are only guaranteed within sequences and not between sequences. Repeated initialization will thus render the numbers obtained less rather than more independent.

Random numbers from other distributions may be obtained from the uniform random numbers by the use of transformations, rejection techniques, and for discrete distributions table based methods.

(a) Transformation methods

For a continuous random variable, if the cumulative distribution function (CDF) is $F(x)$ then for a uniform (0,1) random variate u, $y = F^{-1}(u)$ will have CDF $F(x)$. This method is only efficient in a few simple cases such as the exponential distribution with mean μ, in which case $F^{-1}(u) = -\mu \log u$. Other transformations are based on the joint distribution of several random variables. In the bivariate case, if v and w are random variates there may be a function g such that $y = g(v,w)$ has the required distribution; for example, the Student's t-distribution with n degrees of freedom in which v has a Normal distribution, w has a gamma distibution and $g(v,w) = v\sqrt{n/w}$.

(b) Rejection methods

Rejection techniques are based on the ability to easily generate random numbers from a distribution (called the envelope) similar to the distribution required. The value from the envelope distribution is then accepted as a random number from the required distribution with a certain probability; otherwise, it is rejected and a new number is generated from the envelope distribution.

(c) Table search methods

For discrete distributions, if the cumulative probabilities, $P_i = \text{Prob}(x \leq i)$, are stored in a table then, given u from a uniform (0,1) distribution, the table is searched for i such that $P_{i-1} < u \leq P_i$. The returned value i will have the required distribution. The table searching can be made faster by means of an index, see Ripley [4]. The effort required to set up the table and its index may be considerable, but the methods are very efficient when many values are needed from the same distribution.

In addition to random numbers from various distributions, random compound structures can be generated. These include random time series, random matrices and random samples.

The efficiency of a simulation exercise may often be increased by the use of variance reduction methods (see Morgan [3]). It is also worth considering whether a simulation is the best approach to solving the problem. For example, low-dimensional integrals are usually more efficiently calculated by routines in Chapter D01 rather than by Monte Carlo integration.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

3.1 Design of the Chapter

All the generation routines call – directly or indirectly – an internal basic generator, which generates random numbers from a uniform distribution over (0,1). Thus a call to any generation routine will affect all subsequent random numbers produced by any other routine in the chapter. Despite this effect, the values will remain as independent as if the different sequences were produced separately.

Two utility routines are provided to initialize the basic generator:

G05CBF initializes it to a repeatable state, dependent on an integer parameter: two calls of G05CBF with the same parameter-value will result in the same subsequent sequences of random numbers.

G05CCF initializes it to a non-repeatable state, in such a way that different calls of G05CCF, either in the same run or different runs of the program, will almost certainly result in different subsequent sequences of random numbers.

As mentioned in Section 2, it is important to note that the statistical properties of pseudo-random numbers are only guaranteed within sequences and not between sequences. Repeated initialization will thus render the numbers obtained less rather than more independent. In a simple case there should be only one call to G05CBF or G05CCF, which should be before any call to an actual generation routine.

Two other utility routines, G05CFF and G05CGF, are provided to save or restore the state of the basic generator (including the seed of the multiplicative congruential method used by the basic generator). G05CFF and G05CGF can be used to produce two or more sequences of numbers simultaneously, where some are repeatable and some are not; for example, this can be used to simulate signal and noise. As their overheads are not negligible, numbers should be produced in batches when this technique is used. While they can be used to save the state of the internal generator between jobs, the two arrays must be restored accurately. The corresponding process between machines, while sometimes possible, is not advised.

3.2 Selection of Routine

For three of the commonest continuous distributions – uniform, exponential, and Normal – there is a choice between calling a function to return a single random number and calling a subroutine to fill an array with a sequence of random numbers; the latter is likely to be much more efficient on vector-processing machines.

Distribution	**Function returning a single number**	**Subroutine returning an array of numbers**
uniform over (0,1)	G05CAF	G05FAF
uniform over (a, b)	G05DAF	G05FAF
exponential	G05DBF	G05FBF
Normal	G05DDF	G05FDF

For two discrete distributions, the uniform and Poisson, there is a choice between routines that use indexed search tables, which are suitable for the generation of many variates from the distribution with the same parameters, and routines that are more efficient in the single call situation when the parameters may be changing.

Distribution	**Single call**	**Set up table**
discrete uniform	G05DYF	G05EBF
Poisson	G05DRF	G05ECF

G05EBF and G05ECF return a reference array which is then used by G05EYF.

The following distributions are also available. Those indicated can return more than one value per call.

(a) Continuous Distributions

Beta distribution (multiple)	G05FEF
Cauchy distribution	G05DFF
Chi-square distribution	G05DHF
F-distribution	G05DKF
Gamma distribution (multiple)	G05FFF
Logistic distribution	G05DCF
Lognormal distribution	G05DEF
Student's t-distribution	G05DJF
von Mises distribution	G05FSF
Weibull distribution	G05DPF

(b) Multivariate Distributions

Multivariate Normal distribution	G05EAF and G05EZF

(c) Discrete Distributions using table search

Binomial distribution	G05EDF
Hypergeometric distribution	G05EFF
Negative binomial distribution	G05EEF
User-supplied distribution	G05EXF

The above routines set up the table and index in a reference array; G05EYF can then be called to generate the random variate from the information in the reference array.

(d) Generation of Time Series

Univariate ARMA model, Normal errors	G05EGF and G05EWF
Vector ARMA model, Normal errors	G05HDF

(e) Sampling and Permutation

Random permutation of an integer vector	G05EHF
Random sample from an integer vector	G05EJF
Random logical value	G05DZF

(f) Random Matrices

Random orthogonal matrix	G05GAF
Random correlation matrix	G05GBF

3.3 Programming Advice

Take care when programming calls to those routines in this chapter which are functions. The reason is that different calls with the same parameters are intended to give different results.

For example, if you wish to assign to Z the difference between two successive random numbers generated by G05CAF, beware of writing

```
      Z = G05CAF(X) - G05CAF(X)
```

It is quite legitimate for a Fortran compiler to compile zero, one or two calls to G05CAF; if two calls, they may be in either order (if zero or one calls are compiled, Z would be set to zero). A safe method to program this would be

```
    X = G05CAF(X)
    Y = G05CAF(Y)
    Z = X-Y
```

Another problem that can occur is that an optimising compiler may move a call to a function out of a loop. Thus, the same value would be used for all iterations of the loop, instead of a different random number being generated at each iteration. If this problem occurs, consult an expert on your Fortran compiler.

All the routines in this chapter rely on information stored in common blocks, which must be saved between calls. This need not be a matter of concern unless a program is split into overlays; in such a case, the safest course is to ensure that the G05 routines are in the root overlay.

4 Routines Withdrawn or Scheduled for Withdrawal

Since Mark 13 the following routines have either been withdrawn or superseded. Those routines indicated by a dagger are still present at Mark 18, but will be omitted at a future date. Advice on replacing calls to these routines is given in the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines'.

G05DGF G05DLF G05DMF

5 References

[1] Dagpunar J (1988) *Principles of Random Variate Generation* Oxford University Press

[2] Knuth D E (1981) *The Art of Computer Programming (Volume 2)* Addison-Wesley (2nd Edition)

[3] Morgan B J T (1984) *Elements of Simulation* Chapman and Hall

[4] Ripley B D (1987) *Stochastic Simulation* Wiley

G05CAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05CAF returns a pseudo-random number taken from a uniform distribution between 0 and 1.

2. Specification

```
real FUNCTION G05CAF (X)
real          X
```

3. Description

This routine returns the next pseudo-random number from the basic uniform (0,1) generator.

The basic generator uses a multiplicative congruential algorithm:

$$b_{i+1} = 13^{13} \times b_i \bmod 2^{59}$$

The integer b_{i+1} is divided by 2^{59} to yield a ***real*** value y, which is guaranteed to satisify

$$0 < y < 1.$$

The value of b_i is saved internally in the code. The initial value b_0 is set by default to $123456789 \times (2^{32}+1)$, but the sequence may be re-initialized by a call to G05CBF (for a repeatable sequence) or G05CCF (for a non-repeatable sequence). The current value of b_i may be saved by a call to G05CFF, and restored by a call to G05CGF.

G05FAF may be used to generate a vector of n pseudo-random numbers which are exactly the same as n successive values of G05CAF. On vector-processing machines G05FAF is likely to be much faster.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

5. Parameters

1: X – ***real***. *Dummy*

A dummy argument (originally required by ANSI Fortran 66 syntax).

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

The period of the basic generator is 2^{57}.

Its performance has been analysed by the Spectral Test, see Knuth [1], Section 3.3.4, yielding the following results in the notation of Knuth [1].

n	v_n	Upper bound for v_n
2	3.44×10^8	4.08×10^8
3	4.29×10^5	5.88×10^5
4	1.72×10^4	2.32×10^4
5	1.92×10^3	3.33×10^3
6	593	939
7	198	380
8	108	197
9	67	120

The right hand column gives an upper bound for the values of v_n attainable by any multiplicative congruential generator working modulo 2^{59}.

An informal interpretation of the quantities v_n is that consecutive n-tuples are statistically uncorrelated to an accuracy of $1/v_n$. This is a theoretical result; in practice the degree of randomness is usually much greater than the above figures might support. More details are given in Knuth [1], and in the references cited therein.

Note that the achievable accuracy drops rapidly as the number of dimensions increases. This is a property of all multiplicative congruential generators and is the reason why very long periods are needed even for samples of only a few random numbers.

9. Example

The example program prints the first five pseudo-random numbers from a uniform distribution between 0 and 1, generated by G05CAF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05CAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05CAF
      EXTERNAL         G05CAF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05CAF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05CAF(X)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05CAF Example Program Results

     0.7951
     0.2257
     0.3713
     0.2250
     0.8787
```

G05CBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05CBF sets the seed used by the basic generator in the G05 Chapter to a repeatable initial value.

2. Specification

```
SUBROUTINE G05CBF (I)
INTEGER         I
```

3. Description

This routine sets the internal seed used by the basic generator (see G05CAF) to a value n_0 calculated from the parameter i:

$$n_0 = 2i + 1.$$

It then generates the value n_1 and discards it.

This routine will yield different subsequent sequences of random numbers if called with different values of i, but the sequences will be repeatable in different runs of the calling program. It should be noted that there is no guarantee of statistical properties between sequences, only within sequences.

4. References

None.

5. Parameters

1: I – INTEGER. *Input*

On entry: a number from which the new seed is to be calculated.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a uniform distribution between 0 and 1, generated by G05CAF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05CBF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05CAF
      EXTERNAL         G05CAF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05CBF Example Program Results'
      WRITE (NOUT,*)
*
      CALL G05CBF(0)
*
      DO 20 I = 1, 5
         X = G05CAF(X)
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05CBF Example Program Results

     0.7951
     0.2257
     0.3713
     0.2250
     0.8787
```

G05CCF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05CCF sets the seed used by the basic generator in the G05 Chapter to a non-repeatable initial value.

2. Specification

```
      SUBROUTINE G05CCF
```

3. Description

This routine sets the internal seed used by the basic generator (see G05CAF) to a value n_0 calculated from the setting of the real-time clock. It then generates the value n_1 and discards it.

This routine will yield different subsequent sequences of random numbers in different runs of the calling program. It should be noted that there is no quarantee of statistical properties between sequences, only within sequences.

4. References

None.

5. Parameters

None.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a uniform distribution between 0 and 1, generated by G05CAF after initialisation by G05CCF. The program should give **different** results each time it is run.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05CCF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05CAF
      EXTERNAL         G05CAF
*     .. External Subroutines ..
      EXTERNAL         G05CCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05CCF Example Program Results'
      WRITE (NOUT,*)
*
      CALL G05CCF
*
      DO 20 I = 1, 5
         X = G05CAF(X)
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05CCF Example Program Results

     0.3991
     0.5160
     0.1130
     0.6615
     0.6976
```

G05CFF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05CFF saves the value of the seed used by the basic generator in the G05 Chapter.

2. Specification

```
SUBROUTINE G05CFF (IA, NI, XA, NX, IFAIL)
INTEGER        IA(NI), NI, NX, IFAIL
real           XA(NX)
```

3. Description

This routine saves information about the basic generator to enable G05CGF subsequently to restore the basic generator to its current state. The values of NI, NX, IA and XA must not be altered between a call of G05CFF and a call of G05CGF.

4. References

None.

5. Parameters

1: IA(NI) – INTEGER array. *Output*

On exit: information about the generator.

2: NI – INTEGER. *Input*

On entry: the dimension of the array IA as declared in the (sub)program from which G05CFF is called.

Constraint: NI $\geq$ 9.

3: XA(NX) – ***real*** array. *Output*

On exit: information about the generator.

4: NX – INTEGER. *Input*

On entry: the dimension of the array XA as declared in the (sub)program from which G05CFF is called.

Constraint: NX $\geq$ 4.

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, NI < 9.

IFAIL = 2

On entry, NX < 4.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints 10 pseudo-random numbers generated by G05CAF; it saves the generator state after the 2nd, and restores it after the 7th so that the 8th, 9th and 10th numbers are the same as the 3rd, 4th and 5th.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05CFF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             R
      INTEGER          I, IFAIL
*     .. Local Arrays ..
      real             X(5), XA(4)
      INTEGER          IA(9)
*     .. External Functions ..
      real             G05CAF
      EXTERNAL         G05CAF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05CFF, G05CGF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05CFF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
      DO 20 I = 1, 5
         X(I) = G05CAF(R)
*
         IF (I.EQ.2) CALL G05CFF(IA,9,XA,4,IFAIL)
*
   20 CONTINUE
      WRITE (NOUT,99999) (X(I),I=1,5)
      DO 40 I = 1, 5
         X(I) = G05CAF(R)
*
         IF (I.EQ.2) CALL G05CGF(IA,9,XA,4,IFAIL)
*
   40 CONTINUE
      WRITE (NOUT,99999) (X(I),I=1,5)
      STOP
*
99999 FORMAT (1X,5F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
GO5CFF Example Program Results

     0.7951     0.2257     0.3713     0.2250     0.8787
     0.0475     0.1806     0.3713     0.2250     0.8787
```

G05CGF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05CGF restores the value of the seed used by the basic generator in Chapter G05 after a previous call to G05CFF.

2. Specification

```
SUBROUTINE G05CGF (IA, NI, XA, NX, IFAIL)
INTEGER         IA(NI), NI, NX, IFAIL
real            XA(NX)
```

3. Description

This routine restores the state of the basic generator, using information saved by a previous call to G05CFF.

4. References

None.

5. Parameters

1: IA(NI) – INTEGER array. *Input*

On entry: information about the generator, which must be unchanged from the previous call to G05CFF.

2: NI – INTEGER. *Input*

On entry: the dimension of the array IA as declared in the (sub)program from which G05CGF is called.

Constraint: NI $\geq$ 9

3: XA(NX) – ***real*** array. *Input*

On entry: information about the generator, which must be unchanged from the previous call of G05CFF.

4: NX – INTEGER. *Input*

On entry: the dimension of the array XA as declared in the (sub)program from which G05CGF is called.

Constraint: NX $\geq$ 4

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, NI < 9,
or NX < 4.

IFAIL = 2

On entry, IA or XA has been corrupted since the previous call to G05CFF.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints 10 pseudo-random numbers generated by G05CAF; it saves the generator state after the 2nd, and restores it after the 7th, so that the 8th, 9th and 10th numbers are the same as the 3rd, 4th and 5th.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05CGF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             R
      INTEGER          I, IFAIL
*     .. Local Arrays ..
      real             X(5), XA(4)
      INTEGER          IA(9)
*     .. External Functions ..
      real             G05CAF
      EXTERNAL         G05CAF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05CFF, G05CGF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05CGF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
      DO 20 I = 1, 5
         X(I) = G05CAF(R)
*
         IF (I.EQ.2) CALL G05CFF(IA,9,XA,4,IFAIL)
*
   20 CONTINUE
      WRITE (NOUT,99999) (X(I),I=1,5)
      DO 40 I = 1, 5
         X(I) = G05CAF(R)
*
         IF (I.EQ.2) CALL G05CGF(IA,9,XA,4,IFAIL)
*
   40 CONTINUE
      WRITE (NOUT,99999) (X(I),I=1,5)
      STOP
*
99999 FORMAT (1X,5F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05CGF Example Program Results

     0.7951    0.2257    0.3713    0.2250    0.8787
     0.0475    0.1806    0.3713    0.2250    0.8787
```

G05DAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DAF returns a pseudo-random real number taken from a uniform distribution over the interval $[a,b]$.

2. Specification

```
real FUNCTION G05DAF (A, B)
real          A, B
```

3. Description

This distribution has PDF (probability density function)

$$f(x) = 1/|a-b| \quad \text{if } x \in [a,b],$$
$$f(x) = 0 \quad \text{otherwise.}$$

The routine returns the value

$$x = a + (b-a)y$$

where y is a pseudo-random number from a uniform distribution over $(0,1)$, generated by G05CAF. The routine ensures that x lies in the closed interval $[a,b]$.

G05FAF may be used to generate a vector of n pseudo-random numbers which are exactly the same as n successive values of G05DAF. On vector-processing machines G05FAF is likely to be much faster.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming (Vol. 2).
Addison-Wesley, 1969, (2nd Edition) 1981.

5. Parameters

1: A – ***real***. *Input*
2: B – ***real***. *Input*

On entry: the end-points a and b of the distribution. It is not necessary that $a < b$.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a uniform distribution between 1.0 and 1.5, generated by G05DAF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05DAF
      EXTERNAL         G05DAF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DAF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05DAF(1.0e0,1.5e0)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DAF Example Program Results

     1.3976
     1.1129
     1.1856
     1.1125
     1.4394
```

G05DBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DBF returns a pseudo-random real number taken from a (negative) exponential distribution with mean a.

2. Specification

```
real FUNCTION G05DBF (A)
real             A
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{1}{a}e^{-x/a} \quad \text{if } x > 0,$$

$$f(x) = 0 \quad \text{otherwise.}$$

The routine returns the value $-a \ln y$, where y is a pseudo-random number from a uniform distribution over (0,1), generated by G05CAF.

G05FBF may be used to generate a vector of n pseudo-random numbers which are exactly the same as n successive values of G05DBF. On vector-processing machines G05FBF is likely to be much faster.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: A – ***real***. *Input*

On entry: the parameter a of the distribution. If A is negative, its absolute value is used.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a negative exponential distribution with mean 2.0, generated by G05DBF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DBF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05DBF
      EXTERNAL         G05DBF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DBF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05DBF(2.0e0)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DBF Example Program Results

     0.4585
     2.9769
     1.9816
     2.9830
     0.2585
```

G05DCF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DCF returns a pseudo-random real number taken from a logistic distribution with mean a and spread b.

2. Specification

real `FUNCTION G05DCF (A, B)`
real `A, B`

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{e^{(x-a)/b}}{b\left(1+e^{(x-a)/b}\right)^2}$$

The routine returns the value

$$a + b \ln\left(\frac{y}{1-y}\right)$$

where y is a pseudo-random number uniformly distributed over $(0,1)$, generated by G05CAF.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: A – ***real***. *Input*

On entry: the mean a, of the distribution.

2: B – ***real***. *Input*

On entry: the spread b, of the distribution, where 'spread' is $\frac{\sqrt{3}}{\pi}\times$ standard deviation. If B is negative, the distribution of the generated numbers – though not the actual sequence – is the same as if the absolute value of B were used.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a logistic distribution with mean 1.0 and spread 1.5, generated by G05DCF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DCF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05DCF
      EXTERNAL         G05DCF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DCF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05DCF(1.0e0,1.5e0)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05DCF Example Program Results

    3.0341
   -0.8490
    0.2099
   -0.8548
    3.9709
```

G05DDF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DDF returns a pseudo-random real number taken from a normal (Gaussian) distribution with mean a and standard deviation b.

2. Specification

```
real FUNCTION G05DDF (A, B)
real              A, B
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{1}{b\sqrt{2\pi}} \exp\left(-\frac{(x-a)^2}{2b^2}\right)$$

The routine uses the method of Brent [3].

G05FDF may be used to generate a vector of n pseudo-random numbers from a Normal distribution, but these are **not** the same as n successive values of G05DDF, because G05FDF uses a different method. However on vector-processing machines G05FDF is likely to be much faster.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, 1969, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

[3] BRENT, R.P.
Algorithm 488.
Comm. A.C.M., p. 704, 1974.

5. Parameters

1: A – ***real***. *Input*

On entry: the mean a, of the distribution.

2: B – ***real***. *Input*

On entry: the standard deviation b, of the distribution. If B is negative, the distribution of the generated numbers – though not the actual sequence – is the same as if the absolute value of B were used.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a normal distribution with mean 1.0 and standard deviation 1.5, generated by G05DDF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DDF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05DDF
      EXTERNAL         G05DDF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DDF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05DDF(1.0e0,1.5e0)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DDF Example Program Results

     1.8045
     2.9393
     3.3701
     0.9602
     3.2751
```

G05DEF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DEF returns a pseudo-random real number taken from a log-normal distribution with parameters a and b.

2. Specification

```
real FUNCTION G05DEF (A, B)
real              A, B
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{1}{bx\sqrt{2\pi}} \exp\left(-\frac{(\ln x - a)^2}{2b^2}\right) \quad \text{if } x > 0,$$

$$f(x) = 0 \quad \text{otherwise,}$$

i.e. $\ln x$ is normally distributed with mean a and standard deviation b. The routine returns the value $\exp y$, where y is generated by G05DDF from a Normal distribution with mean a and standard deviation b.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: A – ***real.*** *Input*

On entry: the mean a, of the distribution of $\ln x$.

2: B – ***real.*** *Input*

On entry: the standard deviation b, of the distribution of $\ln x$. If B is negative, the distribution of the generated numbers – though not the actual sequence – is the same as if the absolute value of B were used.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a log-normal distribution with mean 1.0 and standard deviation 1.5, generated by G05DEF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DEF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05DEF
      EXTERNAL         G05DEF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DEF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05DEF(1.0e0,1.5e0)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DEF Example Program Results

     6.0767
    18.9017
    29.0802
     2.6121
    26.4446
```

G05DFF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DFF returns a pseudo-random real number taken from a Cauchy distribution with median a and semi-interquartile range b.

2. Specification

```
real FUNCTION G05DFF (A, B)
real          A, B
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{1}{\pi b\left(1+\left(\frac{x-a}{b}\right)^2\right)}$$

The routine returns the value

$$a + b\frac{2y_1 - 1}{y_2},$$

where y_1 and y_2 are a pair of consecutive pseudo-random numbers from a uniform distribution over (0,1), generated by G05CAF, such that

$$(2y_1-1)^2 + y_2^2 \leq 1.$$

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: A – ***real***. *Input*

On entry: the median a, of the distribution.

2: B – ***real***. *Input*

On entry: the semi-interquartile range b, of the distribution. If B is negative, the distribution of the generated numbers – though not the actual sequence – is the same as if the absolute value of B were used.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random real numbers from a Cauchy distribution with median 1.0 and semi-interquartile range 1.5, generated by G05DFF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DFF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I
*     .. External Functions ..
      real             G05DFF
      EXTERNAL         G05DFF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DFF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05DFF(1.0e0,1.5e0)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DFF Example Program Results

     4.9225
    -0.7160
    24.9342
    -1.2143
     1.6063
```

G05DHF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

Warning: the algorithm for this routine was changed at Mark 16; sequences of numbers produced by earlier versions of this routine will not be repeatable.

1. Purpose

G05DHF returns a pseudo-random real number taken from a χ^2 distribution with n degrees of freedom.

2. Specification

```
real FUNCTION G05DHF (N, IFAIL)
INTEGER       N, IFAIL
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{x^{\frac{1}{2}n-1}\times e^{-x/2}}{2^{\frac{1}{2}n}\times(\frac{1}{2}n-1)!} \quad \text{if } x > 0,$$

$$f(x) = 0 \quad \text{otherwise.}$$

This is the same as a gamma distribution with parameters $\frac{1}{2}n$ and 2; the routine calls G05FFF with these parameters.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of degrees of freedom, n, of the distribution.

Constraint: N $\geq$ 1.

2: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 1.

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine increases with n.

9. Example

The example program prints the first five pseudo-random real numbers from a χ^2 distribution with 5 degrees of freedom, generated by G05DHF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DHF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I, IFAIL
*     .. External Functions ..
      real             G05DHF
      EXTERNAL         G05DHF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DHF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
      DO 20 I = 1, 5
*
         X = G05DHF(5,IFAIL)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DHF Example Program Results

     6.7995
     1.6156
     9.0290
     2.2949
     3.7902
```

G05DJF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

Warning: the algorithm for this routine was changed at Mark 16; sequences of numbers produced by earlier versions of this routine will not be repeatable.

1. Purpose

G05DJF returns a pseudo-random real number taken from a Student's t-distribution with n degrees of freedom.

2. Specification

```
real FUNCTION G05DJF (N, IFAIL)
INTEGER        N, IFAIL
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{\left(\frac{n-1}{2}\right)!}{(\frac{1}{2}n-1)!\sqrt{\pi n}\left(1+\frac{x^2}{n}\right)^{\frac{1}{2}(n+1)}}.$$

The routine returns the value

$$y\sqrt{\frac{n}{z}}$$

where y is generated by G05DDF from a Normal distribution with mean 0 and standard deviation 1.0, and z is generated by G05FFF from a gamma distribution with parameters $\frac{1}{2}n$ and 2 (i.e. from a χ^2 distribution with n degrees of freedom).

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of degrees of freedom, n, of the distribution.

Constraint: $N \geq 1$

2: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, $N < 1$.

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine increases with n.

9. Example

The example program prints the first five pseudo-random real numbers from a Student's t-distribution with 5 degrees of freedom, generated by G05DJF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DJF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I, IFAIL
*     .. External Functions ..
      real             G05DJF
      EXTERNAL         G05DJF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DJF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
      DO 20 I = 1, 5
*
         X = G05DJF(5,IFAIL)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DJF Example Program Results

     0.9435
     1.3828
    -0.4164
     1.0801
     1.1445
```

G05DKF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

Warning: the algorithm for this routine was changed at Mark 16; sequences of numbers produced by earlier versions of this routine will not be repeatable.

1. Purpose

G05DKF returns a pseudo-random real number taken from the F (or Fisher's variance ratio) distribution with m and n degrees of freedom.

2. Specification

```
real FUNCTION G05DKF (M, N, IFAIL)
INTEGER        M, N, IFAIL
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{\left(\frac{m+n-2}{2}\right)!\ x^{\frac{1}{2}m-1}}{(\frac{1}{2}m-1)!\ (\frac{1}{2}n-1)!\ \left(1+\frac{m}{n}x\right)^{\frac{1}{2}(m+n)}} \times \left(\frac{m}{n}\right)^{\frac{1}{2}m} \qquad \text{if } x > 0,$$

$$f(x) = 0 \qquad \text{otherwise.}$$

The routine returns the value

$$\frac{ny}{mz}$$

where y and z are generated by G05FFF from gamma distributions with parameters $(\frac{1}{2}m,2)$ and $(\frac{1}{2}n,2)$ respectively (i.e. from χ^2 distributions with m and n degrees of freedom).

4. References

[1] KNUTH, D.E.
The Art of Computer Programming (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

5. Parameters

1: M – INTEGER. *Input*

On entry: the first degree of freedom, m, of the distribution.

Constraint: M > 0.

2: N – INTEGER. *Input*

On entry: the second degree of freedom, n, of the distribution.

Constraint: N > 0.

3: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, M < 1.

IFAIL = 2

On entry, B $\le$ 0.0.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints out the first five pseudo-random real numbers from a Weibull distribution with shape parameter 1.0 and scale parameter 2.0, generated by G05DPF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DPF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             X
      INTEGER          I, IFAIL
*     .. External Functions ..
      real             G05DPF
      EXTERNAL         G05DPF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DPF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
      DO 20 I = 1, 5
*
         X = G05DPF(1.0e0,2.0e0,IFAIL)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05DPF Example Program Results

     0.4585
     2.9769
     1.9816
     2.9830
     0.2585
```

G05DRF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DRF returns a pseudo-random integer variate from a Poisson distribution with mean λ.

2. Specification

```
INTEGER FUNCTION G05DRF (ALAMDA, IFAIL)
INTEGER          IFAIL
real             ALAMDA
```

3. Description

The distribution of a Poisson random variable X is given by

$$P(X=x) = \frac{\lambda^x e^{-\lambda}}{x!} \quad \text{if } x = 0,1,2,...$$
$$P(X=x) = 0 \quad \text{otherwise.}$$

The methods used by G05DRF have low set-up times and are designed for efficient use when the value of the parameter λ changes during the simulation. For large samples from a distribution with fixed λ using G05ECF to set up a reference vector for subsequent use by G05EYF may be more efficient.

When $\lambda < 7.5$ the product of uniforms method is used, see for example Dagpunar [2]. For larger values of λ an envelope rejection method is used with a target distribution:

$$f(x) = \tfrac{1}{3} \quad \text{if } |x| \leq 1$$
$$f(x) = \tfrac{1}{3}|x|^{-3} \quad \text{otherwise.}$$

This distribution is generated using a ratio of uniforms method. A similar approach has also been suggested by Ahrens and Dieter [1]. The basic method is combined with quick acceptance and rejection tests given by MacLaren [3]. For values of $\lambda \geq 87$ Stirling's approximation is used in the computation of the Poisson distribution function, otherwise tables of factorials are used as suggested by MacLaren [3].

4. References

[1] AHRENS, J.H and DIETER, U.
A Convenient Sampling Method with Bounded Computation Times for Poisson Distributions.
Am. J. Math. Man. Sci., pp. 1-13, 1989.

[2] DAGPUNAR, J.
Principles of Random Variate Generation.
Oxford University Press, 1988.

[3] MACLAREN, N.M.
A Poisson Random Number Generator.
Personal Communication, 1990.

5. Parameters

1: ALAMDA – ***real.*** *Input*

On entry: the parameter, λ, of the distribution.

Constraint: ALAMDA > 0.0 and 2×ALAMDA ≤ MAXINT, where MAXINT is the largest integer representable on the machine (see X02BBF).

2: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

If on exit IFAIL $\neq$ 0 G05DRF will return 0.

IFAIL = 1

On entry, ALAMDA $\leq$ 0.0.

IFAIL = 2

On entry, 2×ALAMDA > MAXINT.

7. Accuracy

Not applicable.

8. Further Comments

The methods used by G05DRF have low set-up times and are designed for efficient use when the value of the parameter λ changes during the simulation. For large samples from a distribution with fixed λ using G05ECF to set up a reference vector for subsequent use by G05EYF may be more efficient.

9. Example

The example program prints five pseudo-random variates from Poisson distributions with parameters 1, 5, 15, 50 and 100 respectively, generated by G05DRF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             A
      INTEGER          IFAIL, X
*     .. External Functions ..
      INTEGER          G05DRF
      EXTERNAL         G05DRF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DRF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
      IFAIL = 0
      CALL G05CBF(0)
   20 READ (NIN,*,END=40) A
*
```

```
      X = G05DRF(A,IFAIL)
*
      WRITE (NOUT,99999) X
      GO TO 20
   40 STOP
*
99999 FORMAT (1X,I10)
      END
```

9.2. Program Data

```
G05DRF Example Program Data
1.0
5.0
15.0
50.0
100.0
```

9.3. Program Results

```
 G05DRF Example Program Results

          1
          3
         14
         51
        121
```

G05DYF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DYF returns a pseudo-random integer taken from a uniform distribution over the interval $[m,n]$.

2. Specification

```
INTEGER FUNCTION G05DYF (M, N)
INTEGER          M, N
```

3. Description

The distribution of a uniform random variable, I, is given by

$$P(I{=}i) = \frac{1}{n-m+1} \quad \text{if } m \le i \le n,$$

$$P(I{=}i) = 0 \quad \text{otherwise,}$$

assuming $m \le n$. The routine returns the value $m + [(n{-}m{+}1)y]$ where [ ] denotes the integer part, and y is a pseudo-random number from a uniform distribution over $(0,1)$, generated by G05CAF. If $m > n$, the roles of m and n are reversed.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

5. Parameters

1: M – INTEGER. *Input*

2: N – INTEGER. *Input*

On entry: the end-points m and n of the distribution. It is not necessary that $m < n$.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random integers from a uniform distribution between –5 and 5, generated by G05DYF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DYF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IX
*     .. External Functions ..
      INTEGER          G05DYF
      EXTERNAL         G05DYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DYF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         IX = G05DYF(-5,5)
*
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DYF Example Program Results

     3
    -3
    -1
    -3
     4
```

G05DZF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05DZF returns a pseudo-random logical value – *true* with probability p and *false* with probability $(1-p)$.

2. Specification

```
LOGICAL FUNCTION G05DZF (P)
```
real P

3. Description

The routine returns the logical value of the relation

$$y < p$$

where y is a pseudo-random number from a uniform distribution over (0,1), generated by G05CAF.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

5. Parameters

1: P – ***real***. *Input*

On entry: the parameter p of the distribution (i.e. the probability of a *true* value). If $p < 0$, the value 0 is used; if $p > 1$, the value 1 is used.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints the first five pseudo-random logical values generated by G05CAF after initialisation by G05CBF, when the probability of a TRUE value is 0.6.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05DZF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I
      LOGICAL          X
*     .. External Functions ..
      LOGICAL          G05DZF
      EXTERNAL         G05DZF
*     .. External Subroutines ..
      EXTERNAL         G05CBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05DZF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      DO 20 I = 1, 5
*
         X = G05DZF(0.6e0)
*
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,L5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05DZF Example Program Results

     F
     T
     T
     T
     F
```

G05EAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EAF sets up a reference vector for a multivariate Normal distribution with mean vector a and covariance matrix C, so that G05EZF may be used to generate pseudo-random vectors.

2. Specification

```
SUBROUTINE G05EAF (A, N, C, IC, EPS, R, NR, IFAIL)
INTEGER        N, IC, NR, IFAIL
real           A(N), C(IC,N), EPS, R(NR)
```

3. Description

When the covariance matrix is non-singular (i.e. strictly positive-definite), the distribution has probability density function

$$f(x) = \sqrt{\frac{|C^{-1}|}{(2\pi)^n}}\exp\{-(x-a)^TC^{-1}(x-a)\}$$

where n is the number of dimensions, C is the covariance matrix, a is the vector of means and x is the vector of positions.

Covariance matrices are symmetric and positive semi-definite. Given such a matrix C, there exists a lower triangular matrix L such that $LL^T = C$. L is not unique, if C is singular.

G05EAF decomposes C to find such an L. It then stores n, a and L in the reference vector r for later use by G05EZF. G05EZF generates a vector x of independent standard Normal pseudo-random numbers. It then returns the vector $a + Lx$, which has the required multivariate Normal distribution.

It should be noted that this routine will work with a singular covariance matrix C, provided C is positive semi-definite, despite the fact that the above formula for the probability density function is not valid in that case. Wilkinson [2] should be consulted if further information is required.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] WILKINSON, J.H.
The Algebraic Eigenvalue Problem.
Clarendon Press, 1965.

5. Parameters

1: A(N) – ***real*** array. *Input*

On entry: the vector of means, a, of the distribution.

2: N – INTEGER. *Input*

On entry: the number of dimensions, n, of the distribution.

Constraint: $N > 0$.

3: C(IC,N) – ***real*** array. *Input*

On entry: the covariance matrix of the distribution. Only the upper triangle need be set.

4: IC – INTEGER. *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which G05EAF is called.

Constraint: IC $\geq$ N.

5: EPS – ***real***. *Input*

On entry: the maximum error in any element of C, relative to the largest element of C.

Constraint: $0.0 \leq$ EPS $\leq 0.1/$N.

If EPS is less than ***machine precision***, ***machine precision*** is used.

6: R(NR) – ***real*** array. *Output*

On exit: the reference vector for subsequent use by G05EZF.

7: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EAF is called.

Constraint: NR $\geq$ ((N+1)$\times$(N+2))/2.

8: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 1.

IFAIL = 2

On entry, NR < ((N+1)$\times$(N+2))/2.

IFAIL = 3

On entry, IC < N.

IFAIL = 4

On entry, EPS < 0.0,
or EPS > 0.1/N.

IFAIL = 5

The covariance matrix C is not positive semi-definite to accuracy EPS.

7. Accuracy

The maximum absolute error in LL^T, and hence in the covariance matrix of the resulting vectors, is less than $(n\times\max(\text{EPS},\varepsilon)+(n+3)\varepsilon/2)$ times the maximum element of C, where ε is the ***machine precision***. Under normal circumstances, the above will be small compared to sampling error.

8. Further Comments

The time taken by the routine is of order n^3.

It is recommended that the diagonal elements of C should not differ too widely in order of magnitude. This may be achieved by scaling the variables if necessary. The actual matrix decomposed is $C + E = LL^T$, where E is a diagonal matrix with small positive diagonal elements. This ensures that, even when C is singular, or nearly singular, the Cholesky Factor L corresponds to a positive-definite covariance matrix that agrees with C within a tolerance determined by EPS.

9. Example

The example program prints five pseudo-random observations from a bivariate Normal distribution with means vector

$$\begin{bmatrix} 1.0 \\ 2.0 \end{bmatrix}$$

and covariance matrix

$$\begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix},$$

generated by G05EAF and G05EZF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N, IC, NR
      PARAMETER        (N=2,IC=N,NR=(N+1)*(N+2)/2)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J
*     .. Local Arrays ..
      real             A(N), C(IC,N), R(NR), Z(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EAF, G05EZF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EAF Example Program Results'
      WRITE (NOUT,*)
      A(1) = 1.0e0
      A(2) = 2.0e0
      C(1,1) = 2.0e0
      C(2,2) = 3.0e0
      C(1,2) = 1.0e0
      C(2,1) = 1.0e0
      CALL G05CBF(0)
      IFAIL = 0
*
      CALL G05EAF(A,N,C,IC,0.01e0,R,NR,IFAIL)
*
      DO 20 I = 1, 5
         IFAIL = 0
         CALL G05EZF(Z,N,R,NR,IFAIL)
         WRITE (NOUT,99999) (Z(J),J=1,N)
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,2F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EAF Example Program Results

     1.7697    4.4481
     3.2678    3.0583
     3.1769    2.3651
    -0.1055    1.8395
     1.2933   -0.1850
```

G05EBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EBF sets up the reference vector, R, for a discrete uniform distribution over the interval $[m,n]$.

2. Specification

```
SUBROUTINE G05EBF (M, N, R, NR, IFAIL)
INTEGER        M, N, NR, IFAIL
real           R(NR)
```

3. Description

This sets up a reference vector for use in G05EYF. Together these routines produce random numbers from the distribution defined by:

$$P(I=i) = \frac{1}{n-m+1} \quad \text{if } m \le i \le n,$$
$$P(I=i) = 0 \quad \text{otherwise,}$$

assuming $m \le n$. If $m > n$, the roles of m and n are reversed.

The reference array is formed in the natural manner (described in more detail in G05EXF).

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, 1969, (2nd Edition) 1981.

5. Parameters

1: M – INTEGER. *Input*

2: N – INTEGER. *Input*

On entry: the endpoints m and n of the distribution. It is not necessary that $m < n$.

3: R(NR) – ***real*** array. *Output*

On exit: the reference vector R.

4: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EBF is called.

Suggested value: approximately $5 + 1.4\times|M-N|$ (for optimum efficiency in G05EYF).

Constraint: $NR > |M-N| + 3$.

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, NR ≤ |M−N| + 3.

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine increases with NR.

9. Example

The example program sets up a reference vector for a uniform distribution between −5 and 5, and then prints the first five pseudo-random numbers generated by G05EYF, after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EBF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          M, N, NR
      PARAMETER        (M=-5,N=5,NR=19)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IX
*     .. Local Arrays ..
      real             R(NR)
*     .. External Functions ..
      INTEGER          G05EYF
      EXTERNAL         G05EYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EBF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
*
      CALL G05EBF(M,N,R,NR,IFAIL)
*
      DO 20 I = 1, 5
         IX = G05EYF(R,NR)
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05EBF Example Program Results

   3
  -3
  -1
  -3
   4
```

G05ECF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05ECF sets up the reference vector R for a Poisson distribution with mean t.

2. Specification

```
SUBROUTINE G05ECF (T, R, NR, IFAIL)
INTEGER        NR, IFAIL
real           T, R(NR)
```

3. Description

This sets up a reference vector for use in G05EYF. Together these routines produce random numbers from the distribution defined by:

$$P(I{=}i) = \frac{t^i e^{-t}}{i!} \quad \text{if } i = 0,1,...$$

$$P(I{=}i) = 0 \quad \text{otherwise.}$$

The reference array is found using a recurrence relation if t is less than 50 and by Stirling's formula otherwise.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: T – ***real***. *Input*

On entry: the mean, t, of the distribution.

Constraint: T $\geq$ 0.

2: R(NR) – ***real*** array. *Output*

On exit: the reference vector.

3: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05ECF is called.

Suggested value: approximately $20 + 20\times\sqrt{T}$ (for optimum efficiency in G05EYF).

Constraint: NR > $(\mathrm{INT}[T+7.15\sqrt{T}+8.5]-\max(0,\mathrm{INT}[T-7.15\sqrt{T}])+4)$.

4: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, T < 0.

IFAIL = 2

On entry, NR is too small (see Section 5).

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine increases with NR.

9. Example

The example program sets up a reference for a Poisson distribution with mean 2.7 and then prints the first five pseudo-random numbers generated by G05EYF, after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05ECF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      real             T
      INTEGER          NR
      PARAMETER        (T=2.7e0,NR=30)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IX
*     .. Local Arrays ..
      real             R(NR)
*     .. External Functions ..
      INTEGER          G05EYF
      EXTERNAL         G05EYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05ECF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05ECF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
*
      CALL G05ECF(T,R,NR,IFAIL)
*
      DO 20 I = 1, 5
         IX = G05EYF(R,30)
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05ECF Example Program Results

     4
     1
     2
     1
     5
```

G05EDF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EDF sets up the reference vector R for a binomial distribution of the number of successes in n trials, each with probability of success p.

2. Specification

```
SUBROUTINE G05EDF (N, P, R, NR, IFAIL)
INTEGER        N, NR, IFAIL
real           P, R(NR)
```

3. Description

G05EDF sets up a reference vector for use in G05EYF. Together these routines produce random numbers from the distribution defined by:

$$P(I=i) = \frac{n!}{i!(n-i)!}p^i(1-p)^{n-i} \qquad \text{if } i = 0,\dots,n,$$
$$P(I=i) = 0 \qquad \text{otherwise.}$$

The reference array is found by a recurrence relation if $np(1-p) < 50$; otherwise Stirling's approximation is used.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of trials, n, of the distribution.

Constraint: $N \geq 0$.

2: P – ***real***. *Input*

On entry: the probability of success, p, of the distribution.

Constraint: $0 \leq P \leq 1$.

3: R(NR) – ***real*** array. *Output*

On exit: the reference vector.

4: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EDF is called.

Suggested value: $NR = 20 + 20\sqrt{N\times P(1-P)}$ approximately (for optimum efficiency in G05EYF).

Constraint: $NR > \min(N,\mathrm{INT}[N\times P+7.15\sqrt{N\times P(1-P)}+1])$
$- \max(0,\mathrm{INT}[N\times P-7.15\sqrt{N\times P(1-P)}-7.15]) + 4$.

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 0.

IFAIL = 2

On entry, P < 0,
or P > 1.

IFAIL = 3

On entry, NR is too small (see Section 5).

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine increases with NR.

9. Example

The example program sets up a reference vector for a binomial distribution with $n = 100$ and $p = 0.5$; it then prints the first five pseudo-random numbers generated by G05EYF, after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EDF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N
      real             P
      INTEGER          NR
      PARAMETER        (N=100,P=0.5e0,NR=125)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IX
*     .. Local Arrays ..
      real             R(NR)
*     .. External Functions ..
      INTEGER          G05EYF
      EXTERNAL         G05EYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EDF Example Program Results'
      WRITE (NOUT,*)
*
      CALL G05CBF(0)
*
      IFAIL = 0
*
```

```
      CALL G05EDF(N,P,R,NR,IFAIL)
*
      DO 20 I = 1, 5
         IX = G05EYF(R,NR)
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EDF Example Program Results

    54
    46
    48
    46
    56
```

G05EEF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EEF sets up the reference vector R for a negative binomial distribution of the number of successes before n failures, where each trial has probability of success p.

2. Specification

```
SUBROUTINE G05EEF (N, P, R, NR, IFAIL)
INTEGER        N, NR, IFAIL
real           P, R(NR)
```

3. Description

G05EEF sets up a reference vector for use in G05EYF. Together these routines produce random numbers from the observation defined by:

$$P(I{=}i) = \frac{(n+i-1)!}{i!(n-1)!}p^i(1-p)^n \qquad \text{if } i = 0,1,\ldots,n,$$

$$P(I{=}i) = 0 \qquad \text{otherwise.}$$

The reference array is generated by a recurrence relation if $np < 50$; otherwise Stirling's approximation is used.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, 1969, (2nd Edition) 1981.

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of failures, n, of the distribution.

Constraint: $N \geq 0$.

2: P – ***real***. *Input*

On entry: the probability of success, p, of the distribution.

Constraint: $0.0 \leq P \leq 1.0$.

3: R(NR) – ***real*** array. *Output*

On exit: the reference vector.

4: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EEF is called.

Suggested value: $NR = 20 + (20\sqrt{N \times P} + 30P)/(1-P)$ approximately (for optimum efficiency in G05EYF).

Constraint: $NR > \mathrm{int}\left(\frac{N \times P + 7.15\sqrt{N \times P} + 20.15}{1-P}\right) - \max\left(0, \frac{N \times P - 7.15\sqrt{N \times P}}{1-P}\right) + 4.$

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 0.

IFAIL = 2

On entry, P < 0.0,
or P > 1.0.

IFAIL = 3

On entry, NR is too small (see Section 5).

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine increases with NR.

9. Example

The example program sets up a reference vector for a negative binomial distribution with $n = 50$ and $p = 0.5$; it then prints the first five pseudo-random numbers generated by G05EYF, after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EEF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N
      real             P
      INTEGER          NR
      PARAMETER        (N=50,P=0.5e0,NR=250)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IX
*     .. Local Arrays ..
      real             R(NR)
*     .. External Functions ..
      INTEGER          G05EYF
      EXTERNAL         G05EYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EEF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
*
```

```
      CALL G05EEF(N,P,R,NR,IFAIL)
*
      DO 20 I = 1, 5
         IX = G05EYF(R,NR)
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EEF Example Program Results

    58
    42
    46
    42
    62
```

G05EFF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EFF sets up a reference vector R for a hypergeometric distribution of the number of specified items in a sample of size l, taken from a population of size n with m specified items in it.

2. Specification

```
SUBROUTINE G05EFF (L, M, N, R, NR, IFAIL)
INTEGER        L, M, N, NR, IFAIL
real           R(NR)
```

3. Description

G05EFF sets up a reference vector for use in G05EYF. Together these routines produce random numbers from the distribution defined by:

$$P(I=i) = \frac{l!m!(n-l)!(n-m)!}{i!(l-i)!(m-i)!(n-m-l+i)!n!} \quad \text{if } i = \max(0,m+l-n),\ldots,\min(l,m),$$

$$P(I=i) = 0 \quad \text{otherwise.}$$

The reference array is generated by a recurrence relation if $lm(n-l)(n-m) < 50n^3$, otherwise Stirling's approximation is used.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

5. Parameters

1: L – INTEGER. *Input*

On entry: the parameter (sample size) l of the distribution.

Constraint: $0 \le L \le N$.

2: M – INTEGER. *Input*

On entry: the parameter (number of specified items) m of the distribution.

Constraint: $0 \le M \le N$.

3: N – INTEGER. *Input*

On entry: the parameter (population size) n of the distribution.

Constraint: $N \ge 0$.

4: R(NR) – ***real*** array. *Output*

On exit: the reference vector.

5: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EFF is called.

Suggested value: $NR = 20 + \sqrt{(L \times M \times (N-M) \times (N-L))/N^3}$ approximately (for optimum efficiency in G05EYF.

Constraint: NR must not be too small, but the limit is too complicated to specify.

6: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N is negative.

IFAIL = 2

On entry, L < 0,
or L > N.

IFAIL = 3

On entry, M < 0,
or M > N.

IFAIL = 4

On entry, NR is too small (see Section 5).

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine increases with NR.

9. Example

The example program sets up a reference vector for a hypergeometric distribution with $l = 100$, $m = 50$ and $n = 1000$; it then prints the first five pseudo-random numbers generated by G05EYF, after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EFF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          L, M, N, NR
      PARAMETER        (L=100,M=50,N=1000,NR=62)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IX
*     .. Local Arrays ..
      real             R(NR)
*     .. External Functions ..
      INTEGER          G05EYF
      EXTERNAL         G05EYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EFF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EFF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
```

```
*
      CALL G05EFF(L,M,N,R,NR,IFAIL)
*
      DO 20 I = 1, 5
         IX = G05EYF(R,NR)
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EFF Example Program Results

     7
     3
     4
     3
     7
```

G05EGF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EGF sets up a reference vector for an autoregressive moving-average (ARMA) time series model with Normally distributed errors, so that G05EWF may be used to generate successive terms. It also initialises the series to a stationary position.

2. Specification

```
SUBROUTINE G05EGF (E, A, NA, B, NB, R, NR, VAR, IFAIL)
INTEGER        NA, NB, NR, IFAIL
real           E, A(*), B(NB), R(NR), VAR
```

3. Description

The ARMA model of such a time series in discrete time is

$$(x_n-E) = A_1(x_{n-1}-E) + \ldots + A_{NA}(x_{n-NA}-E) + B_1\varepsilon_n + \ldots + B_{NB}\varepsilon_{n-NB+1}$$

where x_n is the value of the series at time n, and ε_n is a series of independent random Standard Normal perturbations.

The routine copies A, E and B to the reference vector so that G05EWF can generate the terms of the series. It sets up initial values corresponding to a stationary position using the method described in Tunnicliffe-Wilson [2].

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] TUNNICLIFFE-WILSON, G.
Some Efficient Computational Procedures for High Order ARMA Models.
J. Stat. Comput. Simulation, 8, pp. 301-309, 1979.

5. Parameters

1: E – ***real***. *Input*

On entry: the mean of the time series.

2: A(*) – ***real*** array. *Input*

The dimension of A must be at least max(1,NA).

On entry: the autoregressive coefficients of the model, if any.

3: NA – INTEGER. *Input*

On entry: the number of autoregressive coefficients supplied.

Constraint: NA $\geq$ 0.

4: B(NB) – ***real*** array. *Input*

On entry: the moving-average coefficients of the model.

5: NB – INTEGER. *Input*

On entry: the number of moving-average coefficients supplied.

Constraint: NB $\geq$ 1.

6: R(NR) – ***real*** array. *Output*

On exit: the reference vector and the recent history of the series.

7: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EGF is called.

Constraint: NR ≥ NA + NB + 4 + max(NA,NB).

8: VAR – ***real.*** *Output*

On exit: the proportion of the variance of a term in the series that is due to the moving-average (error) terms in the model. The smaller this is, the nearer is the model to non-stationarity.

9: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, NA < 0.

IFAIL = 2

On entry, NB < 1.

IFAIL = 3

On entry, NR < NA + NB + 4 + max(NA,NB).

IFAIL = 4

A does not define a stationary autoregressive process.

7. Accuracy

The errors in the initialisation process should be very much smaller than the error term; see Tunnicliffe-Wilson [2].

8. Further Comments

The time taken by the routine is essentially of order $(\mathrm{NA})^2$.

Note: G05CBF, G05CCF, G05CFF, and G05CGF must be used with care if this routine is used as well. The reference vector, as mentioned before, contains a copy of the recent history of the series. This will not be altered properly by calls to any of the above routines. A call to G05CBF or G05CCF should be followed by calls to G05EGF to re-initialise all time series reference vectors in use. To maintain repeatability with G05CBF, the calls to G05EGF should be performed in the same order and at the same point or points in the simulation every time G05CBF is used. When routines G05CFF and G05CGF are used to save and restore the generator state, all the time series reference vectors in use must be saved and restored as well.

The ARMA model for a time series can also be written as:

$$\begin{aligned}(x_t-c) &= \phi_1(x_{t-1}-c) + \ldots + \phi_p(x_{t-p}-c)\\ &\quad + a_t - \theta_1 a_{t-1} \ldots - \theta_q a_{t-q}\end{aligned}$$

where

x_t is the observed value of the time series at time t,

p is the number of autoregressive parameters, ϕ_i,
q is the number of moving average parameters, θ_i,
c is the mean of the time series,
and a_t is a series of independent random Normal perturbations with variance σ^2.

This is the form used in Chapter G13. This is related to the form given in Section 3 by:

$$B_1^2 = \sigma^2,$$
$$B_{i+1} = -\theta_i\sigma = -\theta_i B_1, \qquad i = 1,2,...q,$$
$$NB = q + 1,$$
$$E = c,$$
$$A_i = \phi_i \quad i = 1,2,...,p.$$

and NA $= p$.

9. Example

This example program calls G05EGF to set up the reference vector for the autoregressive model

$$x_n = 0.4x_{n-1} + 0.2x_{n-2} + \varepsilon_n$$

where ε_n is a series of independent random Standard Normal perturbations. G05EWF is then called 10 times to generate a sample of observations, which are printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EGF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NA, NB, NR
      PARAMETER        (NA=2,NB=1,NR=NA+NB+4+NA)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             VAR, X
      INTEGER          I, IFAIL
*     .. Local Arrays ..
      real             A(NA), B(NB), R(NR)
*     .. External Functions ..
      real             G05EWF
      EXTERNAL         G05EWF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EGF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EGF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      A(1) = 0.4e0
      A(2) = 0.2e0
      B(1) = 1.0e0
      IFAIL = 0
*
      CALL G05EGF(0.0e0,A,NA,B,NB,R,NR,VAR,IFAIL)
*
      DO 20 I = 1, 10
         IFAIL = 0
         X = G05EWF(R,NR,IFAIL)
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F12.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EGF Example Program Results

        2.4084
        1.1987
        2.4778
        0.7998
        0.0452
        0.4125
        0.3784
       -1.2166
       -0.3510
        1.1631
```

G05EHF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EHF performs a pseudo-random permutation of a vector of integers.

2. Specification

```
SUBROUTINE G05EHF (INDEX, N, IFAIL)
INTEGER        INDEX(N), N, IFAIL
```

3. Description

The routine permutes the elements of INDEX without inspecting their values. Each of the $n!$ possible permutations of the n values may be regarded as being equiprobable.

If n is 20 or more, it is theoretically impossible that all $n!$ permutations may occur, as $n!$ exceeds the cycle length of G05CAF. For practical purposes this is irrelevant, as the time necessary to generate all possible permutations is many millenia.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 2).
Griffin, London, (3rd Edition) 1969.

5. Parameters

1: INDEX(N) – INTEGER array. *Input/Output*

On entry: the n integer values to be permuted.

On exit: the n permuted integer values.

2: N – INTEGER. *Input*

On entry: the number of values to be permuted.

Constraint: N $\geq$ 1.

3: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 1.

7. Accuracy

Not relevant.

8. Further Comments

The time taken by the routine is of order n.

In order to permute other kinds of vectors, or matrices of higher dimension, the following technique may be used:

(a) Set INDEX(i) = i, for i = 1,2,...,n
(b) Use G05EHF to permute INDEX
(c) Use the contents of INDEX as a set of indices to access the relevant vector or matrix.

In order to divide pseudo-randomly a vector or matrix into subgroups of chosen sizes, a similar procedure may be used. INDEX is first set to the number of 1's, 2's, etc., corresponding to the size of each group, then permuted, and used to index the groups.

9. Example

A vector containing the first 8 positive integers in ascending order is permuted and the permutation is printed. This is repeated a total of 10 times.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EHF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N
      PARAMETER        (N=8)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J, K, M
*     .. Local Arrays ..
      INTEGER          INDEX(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EHF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EHF Example Program Results'
      WRITE (NOUT,*)
      M = 10
      CALL G05CBF(0)
      WRITE (NOUT,99998) M, ' Permutations of first ', N, ' integers'
      WRITE (NOUT,*)
      DO 40 J = 1, M
         DO 20 I = 1, N
            INDEX(I) = I
   20    CONTINUE
         IFAIL = 0
*
         CALL G05EHF(INDEX,N,IFAIL)
*
         WRITE (NOUT,99999) (INDEX(K),K=1,N)
   40 CONTINUE
      STOP
*
99999 FORMAT (1X,8I3)
99998 FORMAT (1X,I2,A,I1,A)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05EHF Example Program Results

10 Permutations of first 8 integers

  7   8   1   2   4   6   3   5
  3   1   4   6   7   8   5   2
  7   6   5   1   3   4   8   2
  6   2   7   3   8   5   1   4
  1   6   2   4   7   8   5   3
  4   1   5   8   7   6   3   2
  8   3   1   6   4   2   5   7
  1   2   6   7   8   4   3   5
  2   5   7   6   3   1   4   8
  2   8   6   7   3   5   1   4
```

G05EJF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EJF selects a pseudo-random sample without replacement from an integer vector.

2. Specification

```
SUBROUTINE G05EJF (IA, N, IB, M, IFAIL)
INTEGER        IA(N), N, IB(M), M, IFAIL
```

3. Description

The routine selects m elements from vector IA of length n and places them in vector IB. Their order in IA will be preserved in IB. Each of the $\binom{n}{m}$ possible combinations of elements of IA may be regarded as being equiprobable.

If n is greater than 60, it is theoretically impossible that all combinations of size m may occur, unless m is near 1 or near n. This is because $\binom{n}{m}$ exceeds the cycle length of G05CAF. For practical purposes this is irrelevant, as the time taken to generate all possible combinations is many millenia.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, 1969, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, London, (3rd Edition) 1969.

5. Parameters

1: IA(N) – INTEGER array. *Input*

On entry: the population to be sampled.

2: N – INTEGER. *Input*

On entry: the number of elements in the vector to be sampled.

Constraint: N $\geq$ 1.

3: IB(M) – INTEGER array. *Output*

On exit: the selected sample.

4: M – INTEGER. *Input*

On entry: the sample size.

Constraint: 1 $\leq$ M $\leq$ N.

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 1.

IFAIL = 2

On entry, M < 1,
or M > N.

7. Accuracy

Not relevant.

8. Further Comments

The time taken by the routine is of order n.

In order to sample other kinds of vectors, or matrices of higher dimension, the following technique may be used:

(a) Set IA(i) = i, for $i = 1,2,...,n$
(b) Use G05EJF to take a sample from IA and put it into IB
(c) Use the contents of IB as a set of indices to access the relevant vector or matrix.

In order to divide a population into several groups, G05EHF (q.v.) is more efficient.

9. Example

From a vector containing the first 8 positive integers in ascending order, random samples of size 1,2,...,8 are selected and printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EJF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N
      PARAMETER        (N=8)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, K, M
*     .. Local Arrays ..
      INTEGER          IA(N), IB(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EJF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EJF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      WRITE (NOUT,99999) 'Samples from the first ', N, ' integers'
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Sample size     Values'
      DO 20 I = 1, N
         IA(I) = I
   20 CONTINUE
      DO 40 M = 1, N
         IFAIL = 0
*
```

```
            CALL G05EJF(IA,N,IB,M,IFAIL)
*
            WRITE (NOUT,99998) M, (IB(K),K=1,M)
   40    CONTINUE
         STOP
*
99999 FORMAT (1X,A,I1,A)
99998 FORMAT (1X,I6,10X,8I3)
         END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EJF Example Program Results

 Samples from the first 8 integers

 Sample size     Values
      1            6
      2            1  7
      3            1  3  4
      4            1  2  6  8
      5            1  3  4  6  7
      6            1  2  3  4  5  6
      7            1  2  3  4  6  7  8
      8            1  2  3  4  5  6  7  8
```

G05EWF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EWF returns the next term from an autoregressive moving-average time series using a reference vector set up by G05EGF.

2. Specification

```
real FUNCTION G05EWF (R, NR, IFAIL)
INTEGER          NR, IFAIL
real             R(NR)
```

3. Description

The routine generates the next term in the autoregressive series and stores it in a circular buffer in the reference vector. It then applies the moving-average summation and returns the result. This is equivalent to the ARMA model described under G05EGF.

4. References

[1] TUNNICLIFFE-WILSON, G.
Some Efficient Computational Procedures for High Order ARMA Models.
J. Stat. Comput. Simulation, 8, pp. 301-309, 1979.

5. Parameters

1: R(NR) – ***real*** array. *Input/Output*

On entry: the reference vector as set up by G05EGF.

On exit: the updated reference vector.

2: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EWF is called.

This should be the same as in the preceding call of G05EGF.

3: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

NR has been changed or R corrupted since it was set up by G05EGF, or since its last use by G05EWF.

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine is of order (NA+NB), where NA is the number of autoregressive coefficients, and NB the number of moving-average coefficients, in the underlying model.

The comments made in Section 8 of the document for G05EGF, concerning the use of G05CBF, G05CCF, G05CFF and G05CGF, must be read before using this routine.

Although the reference vector may be copied like any other array, inexperienced users are strongly advised not to keep more than a single copy. Copying it at any point has the effect of starting a new, independent time series with an identical history. This facility may be useful, but it is clearly a fruitful source of confusion if misused or used by accident.

9. Example

This example program calls G05EGF to set up the reference vector for the autoregressive model

$$x_n = 0.4x_{n-1} + 0.2x_{n-2} + \varepsilon_n$$

where ε_n is a series of independent random Standard Normal perturbations. G05EWF is then called 10 times to generate a sample of observations, which are printed.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EWF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NA, NB, NR
      PARAMETER        (NA=2,NB=1,NR=NA+NB+4+NA)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             VAR, X
      INTEGER          I, IFAIL
*     .. Local Arrays ..
      real             A(NA), B(NB), R(NR)
*     .. External Functions ..
      real             G05EWF
      EXTERNAL         G05EWF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EGF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EWF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      A(1) = 0.4e0
      A(2) = 0.2e0
      B(1) = 1.0e0
      IFAIL = 0
*
      CALL G05EGF(0.0e0,A,NA,B,NB,R,NR,VAR,IFAIL)
*
      DO 20 I = 1, 10
         IFAIL = 0
         X = G05EWF(R,NR,IFAIL)
         WRITE (NOUT,99999) X
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,F12.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EWF Example Program Results

        2.4084
        1.1987
        2.4778
        0.7998
        0.0452
        0.4125
        0.3784
       -1.2166
       -0.3510
        1.1631
```

G05EXF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EXF sets up the reference vector R for a discrete distribution with PDF (probability density function) or CDF (cumulative distribution function) P.

2. Specification

```
SUBROUTINE G05EXF (P, NP, IP, LP, R, NR, IFAIL)
INTEGER        NP, IP, NR, IFAIL
real           P(NP), R(NR)
LOGICAL        LP
```

3. Description

G05EXF sets up a reference vector R for use in G05EYF according to information supplied by the user in P. This may either be the PDF or CDF of the distribution. The reference vector contains the CDF of the distribution in its higher elements, preceded by an index of the form:

R(1) = the number of elements of index, k

R(2) = the value of IP – the (possibly non-positive) subscript in R of the element of the CDF corresponding to P(1) [i.e. R(2) $\leq$ IP $-$ $(k+3)$].

R$(i+2)$ = $\min\{j|\mathrm{CDF}(j)>(i-1)/k\}$, $i = 1,2,\ldots,k$

R(i), $i = k+3,\ldots,$NR, the CDF.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: P(NP) – ***real*** array. *Input*

On entry: the PDF or CDF of the distribution.

2: NP – INTEGER. *Input*

On entry: the dimension of the array P as declared in the (sub)program from which G05EXF is called.

Constraint: NP > 0.

3: IP – INTEGER. *Input*

On entry: the value of the variate, assumed to be a whole number, to which the probability in P(1) corresponds.

4: LP – LOGICAL. *Input*

On entry: LP indicates the type of information contained in P. If LP is .TRUE., P contains a cumulative distribution function (CDF); if LP is .FALSE., P contains a probability density function (PDF).

5: R(NR) – ***real*** array. *Output*

On exit: the reference vector R (see Section 3).

6: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EXF is called.

Suggested value: NR = 5 + 1.4×NP approximately (for optimum efficiency in G05EYF).

Constraint: NR > NP + 2.

7: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, NP < 1.

IFAIL = 2

On entry, NR ≤ NP + 2.

IFAIL = 3

If LP is .TRUE. on entry, then the values in P are not all in non-descending order, as required by a CDF. If LP is .FALSE., then at least one of the probabilities in P is negative, or all the probabilities are zero.

IFAIL = 4

The total probability is not 1. In this case, R is set up correctly since the error may be due to larger rounding errors than expected.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program sets up a reference vector for a distribution whose CDF, $f(n)$, is defined as follows:

n	$f(n)$
0	0.0
1	0.1
2	0.2
3	0.4
4	0.5
5	0.6
6	0.8
7	0.9
8	1.0
9	1.0

It then prints the first five pseudo-random numbers generated by G05EXF, after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EXF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NP, NR
      PARAMETER        (NP=10,NR=19)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IX
*     .. Local Arrays ..
      real             P(NP), R(NR)
*     .. External Functions ..
      INTEGER          G05EYF
      EXTERNAL         G05EYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EXF
*     .. Data statements ..
      DATA             P/0.0e0, 0.1e0, 0.2e0, 0.4e0, 0.5e0, 0.6e0,
     +                 0.8e0, 0.9e0, 1.0e0, 1.0e0/
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EXF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
*
      CALL G05EXF(P,NP,0,.TRUE.,R,NR,IFAIL)
*
      DO 20 I = 1, 5
         IX = G05EYF(R,NR)
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05EXF Example Program Results

     6
     3
     3
     3
     7
```

G05EYF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EYF returns a pseudo-random integer taken from a discrete distribution defined by a reference vector R.

2. Specification

```
INTEGER FUNCTION G05EYF (R, NR)
INTEGER          NR
real             R(NR)
```

3. Description

This routine is designed for use in conjunction with other routines in this chapter, which set up the reference vector R for specific distributions or according to a distribution specified in terms of the PDF (probability density function) or CDF (cumulative distribution function). See the G05 Chapter Introduction.

The routine generates a random number x from G05CAF and searches the CDF in R for the smallest value y such that $\mathrm{CDF}(y) \geq x$ and $\mathrm{CDF}(y-1) < x$. R will contain an index to speed this process and the more space allotted to R the faster this will become.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: R(NR) – ***real*** array. *Input*

On entry: the reference vector R.

2: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EYF is called. It must be the same as the value of NR specified in a call to a routine to set up the reference vector.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

The time taken by the routine decreases as the space allotted to the index part of R increases. There is a point, depending on the distribution, where the improvement becomes very small and the recommended values for NR in other routines are designed to approximate this point.

9. Example

The example program calls G05ECF to set up a reference vector for a Poisson distribution with mean 2.7; it then prints the first five pseudo-random numbers generated by G05EYF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EYF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      real             T
      INTEGER          NR
      PARAMETER        (T=2.7e0,NR=30)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IX
*     .. Local Arrays ..
      real             R(NR)
*     .. External Functions ..
      INTEGER          G05EYF
      EXTERNAL         G05EYF
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05ECF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EYF Example Program Results'
      WRITE (NOUT,*)
      CALL G05CBF(0)
      IFAIL = 0
*
      CALL G05ECF(T,R,NR,IFAIL)
*
      DO 20 I = 1, 5
*
         IX = G05EYF(R,NR)
*
         WRITE (NOUT,99999) IX
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,I5)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EYF Example Program Results

     4
     1
     2
     1
     5
```

G05EZF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05EZF generates a pseudo-random multivariate Normal vector taken from a distribution described by a reference vector set up by G05EAF.

2. Specification

```
SUBROUTINE G05EZF (Z, N, R, NR, IFAIL)
INTEGER        N, NR, IFAIL
real           Z(N), R(NR)
```

3. Description

This routine is designed for use in conjunction with G05EAF. The description of G05EAF should be referred to for a specification of the operation of these two routines.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: Z(N) – ***real*** array. *Output*

On exit: the pseudo-random multivariate Normal vector generated by the routine.

2: N – INTEGER. *Input*

On entry: the dimension, n, of the distribution. This must be the same as was set up in the reference vector by G05EAF.

Constraint: N $\geq$ 1.

3: R(NR) – ***real*** array. *Input*

On entry: the reference vector as set up by G05EAF.

4: NR – INTEGER. *Input*

On entry: the dimension of the array R as declared in the (sub)program from which G05EZF is called. It must be the same as the value of NR specified in the call to G05EAF to set up the reference vector.

Constraint: NR $\geq$ (N+1)(N+2)/2.

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

IFAIL = 1

On entry, N < 1,

or N is not the same as when R was set up by G05EAF. This is likely to be due to corruption of R.

IFAIL = 2

On entry, NR < (N+1)(N+2)/2.

7. Accuracy

The accuracy is discussed in the routine document for G05EAF.

8. Further Comments

The time taken by the routine is of the order

$$a + b\times n + c\times n^2$$

where a and b are appreciably (say 10-20 times) larger than c.

9. Example

The example program prints five pseudo-random observations from a bivariate Normal distribution with means vector

$$\begin{bmatrix} 1.0 \\ 2.0 \end{bmatrix}$$

and covariance matrix

$$\begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix}\text{'}$$

generated by G05EAF and G05EZF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05EZF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N, NR, IC
      PARAMETER        (N=2,NR=(N+1)*(N+2)/2,IC=N)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J
*     .. Local Arrays ..
      real             A(N), C(IC,N), R(NR), Z(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05EAF, G05EZF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05EZF Example Program Results'
      WRITE (NOUT,*)
      A(1) = 1.0e0
      A(2) = 2.0e0
      C(1,1) = 2.0e0
      C(2,2) = 3.0e0
      C(1,2) = 1.0e0
      C(2,1) = 1.0e0
      CALL G05CBF(0)
      IFAIL = 0
*
```

```
      CALL G05EAF(A,N,C,IC,0.01e0,R,NR,IFAIL)
*
      DO 20 I = 1, 5
*
         CALL G05EZF(Z,N,R,NR,IFAIL)
*
         WRITE (NOUT,99999) (Z(J),J=1,N)
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,2F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05EZF Example Program Results

     1.7697    4.4481
     3.2678    3.0583
     3.1769    2.3651
    -0.1055    1.8395
     1.2933   -0.1850
```

G05FAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05FAF generates a vector of pseudo-random numbers uniformly distributed over the interval $[a,b]$.

2. Specification

```
SUBROUTINE G05FAF (A, B, N, X)
INTEGER        N
real           A, B, X(N)
```

3. Description

If $a = 0$ and $b = 1$, G05FAF returns the next n values y_i from the basic uniform (0,1) generator (see G05CAF for details).

For other values of a and b, G05FAF applies the transformation

$$x_i = a + (b-a)y_i$$

The routine ensures that the values x_i lie in the closed interval $[a,b]$.

G05FAF always generates exactly the same pseudo-random numbers as would n consecutive calls of G05CAF or G05DAF, and on vector-processing machines is likely to be much faster.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

5. Parameters

1: A – ***real***. *Input*

2: B – ***real***. *Input*

On entry: the end-points a and b of the uniform distribution. It is not necessary that $a < b$.

3: N – INTEGER. *Input*

On entry: the number n of pseudo-random numbers to be generated.

4: X(N) – ***real*** array. *Output*

On exit: the n pseudo-random numbers from the specified uniform distribution.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints 5 pseudo-random numbers from a uniform distribution between 1.0 and 1.5, generated by a single call to G05FAF, after initialization by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05FAF Example Program Text
*     Mark 14 Release.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=5)
*     .. Local Scalars ..
      INTEGER          I
*     .. Local Arrays ..
      real             X(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05FAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05FAF Example Program Results'
      CALL G05CBF(0)
*
      CALL G05FAF(1.0e0,1.5e0,N,X)
*
      WRITE (NOUT,99999) (X(I),I=1,N)
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05FAF Example Program Results
     1.3976
     1.1129
     1.1856
     1.1125
     1.4394
```

G05FBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05FBF generates a vector of pseudo-random numbers taken from a (negative) exponential distribution with mean a.

2. Specification

```
SUBROUTINE G05FBF (A, N, X)
INTEGER        N
real           A, X(N)
```

3. Description

The distribution has PDF (probability density function)

$$f(x) = \frac{1}{a}e^{-x/a} \quad \text{if } x > 0,$$

$$f(x) = 0 \quad \text{otherwise.}$$

The routine returns the values

$$x_i = -a \ln y_i$$

where y_i are the next n numbers generated by the basic uniform (0,1) generator.

G05FBF always generates exactly the same pseudo-random numbers as would n consecutive calls of G05DBF, but on vector-processing machines is likely to be much faster.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: A – ***real***. *Input*

On entry: the parameter a of the distribution. If A is negative, its absolute value is used.

2: N – INTEGER. *Input*

On entry: the number n of pseudo-random numbers to be generated.

3: X(N) – ***real*** array. *Output*

On exit: the n pseudo-random numbers from the specified exponential distribution.

6. Error Indicators and Warnings

None.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints 5 pseudo-random numbers from an exponential distribution with mean 2.0, generated by a single call to G05FBF, after initialization by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05FBF Example Program Text
*     Mark 14 Release.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=5)
*     .. Local Scalars ..
      INTEGER          I
*     .. Local Arrays ..
      real             X(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05FBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05FBF Example Program Results'
      CALL G05CBF(0)
*
      CALL G05FBF(2.0e0,N,X)
*
      WRITE (NOUT,99999) (X(I),I=1,N)
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05FBF Example Program Results
     0.4585
     2.9769
     1.9816
     2.9830
     0.2585
```

G05FDF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05FDF generates a vector of pseudo-random numbers taken from a Normal (Gaussian) distribution with mean a and standard deviation b.

2. Specification

```
SUBROUTINE G05FDF (A, B, N, X)
INTEGER        N
real           A, B, X(N)
```

3. Description

The distribution has PDF (probability distribution function)

$$f(x) = \frac{1}{b\sqrt{2\pi}}\exp\left(-\frac{(x-a)^2}{2b^2}\right).$$

The routine uses the Box-Muller method.

The routine does **not** generate the same pseudo-random numbers as would n consecutive calls of G05DDF, because G05DDF uses a different method. However on vector-processing machines G05FDF is likely to be much faster.

4. References

[1] KNUTH, D.E.
The Art of Computer Programming, (Vol. 2).
Addison-Wesley, (2nd Edition) 1981.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, (Vol. 1).
Griffin, (3rd Edition) 1969.

5. Parameters

1: A – ***real***. *Input*

On entry: the mean, a, of the distribution.

2: B – ***real***. *Input*

On entry: the standard deviation, b, of the distribution. If B is negative, the distribution of the generated numbers – though not the actual sequence – is the same as if the absolute value of B were used.

3: N – INTEGER. *Input*

On entry: the number n of pseudo-random numbers to be generated.

4: X(N) – ***real*** array. *Output*

On exit: the n pseudo-random numbers from the specified Normal distribution.

6. Error Indicators and Warnings

None.

7. Accuracy

The generated numbers conform to a Normal distribution with an accuracy of $\sqrt{\textit{machine precision}}$.

8. Further Comments

None.

9. Example

The example program prints 5 pseudo-random numbers from a Normal distribution with mean 1.0 and standard deviation 1.5, generated by a single call to G05FDF, after initialization by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05FDF Example Program Text
*     Mark 14 Release.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=5)
*     .. Local Scalars ..
      INTEGER          I
*     .. Local Arrays ..
      real             X(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05FDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05FDF Example Program Results'
      CALL G05CBF(0)
*
      CALL G05FDF(1.0e0,1.5e0,5,X)
*
      WRITE (NOUT,99999) (X(I),I=1,N)
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05FDF Example Program Results
     1.1544
     2.0039
     1.3299
     3.0856
     1.7290
```

G05FEF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05FEF generates a vector of pseudo-random variates from a beta distribution with parameters a and b.

2. Specification

```
SUBROUTINE G05FEF (A, B, N, X, IFAIL)
INTEGER        N, IFAIL
real           A, B, X(N)
```

3. Description

The beta distribution has PDF (probability density function):

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1} \qquad \text{if } 0 \le x \le 1;\ a,b > 0.0$$

$$f(x) = 0 \qquad \text{otherwise.}$$

One of four algorithms is used to generate the variates depending on the values of a and b. Let α be the maximum and β be the minimum of a and b. Then the algorithms are as follows:

If $\alpha < 0.5$

Jöhnk's algorithm is used, see for example Dagpunar [3]. This generates the beta variate as $u_1^{1/a}/(u_1^{1/a}+u_2^{1/b})$, where u_1 and u_2 are uniformly distributed random variates.

If $\beta > 1$

The algorithm BB given by Cheng [2] is used. This involves the generation of an observation from a beta distribution of the second kind by the envelope rejection method using a log-logistic target distribution and then transforming it to a beta variate.

If $\alpha > 1$ and $\beta < 1$

The switching algorithm given by Atkinson [1] is used. The two target distributions used are $f_1(x) = \beta x^{\beta}$ and $f_2(x) = \alpha(1-x)^{\beta-1}$, along with the approximation to the switching parameter of $t = (1-\beta)/(\alpha+1-\beta)$.

In all other cases

Cheng's BC algorithm, see [2], is used with modifications suggested by Dagpunar [3]. This algorithm is similar to BB, used when $\beta > 1$, but is tuned for small values of a and b.

4. References

[1] ATKINSON, A.C.
A Family of Switching Algorithms for the Computer Generation of Beta Random Variates.
Biometrika, 66, pp. 141-5, 1979.

[2] CHENG, R.C.H.
Generating Beta Variates with Nonintegral Shape Parameters.
Comm. ACM, Vol. 21, No. 4, pp. 317-322, April, 1978.

[3] DAGPUNAR, J.
Principles of Random Variate Generation.
Oxford University Press, 1988.

[4] HASTINGS, N.A.J. and PEACOCK, J.B.
Statistical Distributions.
Butterworths, 1975.

5. Parameters

1: A – ***real.*** *Input*

On entry: the parameter, a, of the beta distribution.

Constraint: A > 0.0.

2: B – ***real.*** *Input*

On entry: the parameter, b, of the beta distribution.

Constraint: B > 0.0.

3: N – INTEGER. *Input*

On entry: the number, n, of pseudo-random numbers to be generated.

Constraint: N ≥ 1.

4: X(N) – ***real*** array. *Output*

On exit: the n pseudo-random variates from the specified beta distribution.

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, A ≤ 0.0,
or B ≤ 0.0,
or N < 1.

7. Accuracy

Not applicable.

8. Further Comments

To generate an observation, y, from the beta distribution of the second kind from an observation, x, generated by G05FEF the transformation, $y = x/(1-x)$, may be used.

9. Example

The example program prints a set of five pseudo-random variates from a beta distribution with parameters $a = 2.0$ and $b = 2.0$, generated by G05FEF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05FEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=5)
```

```
*     .. Local Scalars ..
      INTEGER          IFAIL, J
*     .. Local Arrays ..
      real             X(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05FEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05FEF Example Program Results'
      WRITE (NOUT,*)
      IFAIL = 0
      CALL G05CBF(0)
      WRITE (NOUT,*) 'Beta Dist --- A=2.0, B=2.0'
*
      CALL G05FEF(2.0e0,2.0e0,N,X,IFAIL)
*
      WRITE (NOUT,99999) (X(J),J=1,N)
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05FEF Example Program Results

 Beta Dist --- A=2.0, B=2.0
     0.7229
     0.4079
     0.8023
     0.2555
     0.0946
```

G05FFF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05FFF generates a vector of pseudo-random variates from a gamma distribution with parameters a and b.

2. Specification

```
SUBROUTINE G05FFF (A, B, N, X, IFAIL)
INTEGER        N, IFAIL
real           A, B, X(N)
```

3. Description

The gamma distribution has PDF (probability density function):

$$f(x) = \frac{1}{b^a\Gamma(a)}x^{a-1}e^{-x/b} \qquad \text{if } 0 \le x;\ a,\ b > 0.0$$

$$f(x) = 0 \qquad \text{otherwise.}$$

One of three algorithms is used to generate the variates depending upon the value of a:

If $a < 1$

A switching algorithm described by Dagpunar [3] (called G6), is used. The target distributions are $f_1(x) = cax^{a-1}/t^a$ and $f_2(x) = (1-c)e^{-(x-t)}$, where $c = t(t+ae^{-t})$, and the switching parameter, t, is taken as $1-a$. This is similar to Ahrens and Dieter's GS algorithm [1] in which $t = 1$.

If $a = 1$

The gamma distribution reduces to the exponential distribution and the method based on the logarithmic transformation of a uniform random variate is used.

If $a > 1$

The algorithm given by Best [2] is used. This is based on using a Student's t-distribution with two degrees of freedom as the target distribution in an envelope rejection method.

4. References

[1] AHRENS, J.H. and DIETER, U.
Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions.
Comput., 12, pp. 223-46, 1974.

[2] BEST, D.J.
Letter to the Editor.
Appl. Statist., 29, p. 181, 1978.

[3] DAGPUNAR, J.
Principles of Random Variate Generation.
Oxford University Press, 1988.

[4] HASTINGS, N.A.J. and PEACOCK, J.B.
Statistical Distributions.
Butterworths, 1975.

5. Parameters

1: A – ***real***. *Input*

On entry: the parameter, a, of the gamma distribution.

Constraint: A > 0.0.

2: B – ***real***. *Input*

On entry: the parameter, b, of the gamma distribution.

Constraint: B > 0.0.

3: N – INTEGER. *Input*

On entry: the number, n, of pseudo-random numbers to be generated.

Constraint: N ≥ 1.

4: X(N) – ***real*** array. *Output*

On exit: the n pseudo-random variates from the specified gamma distribution.

5: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, A ≤ 0.0,
or B ≤ 0.0,
or N < 1.

7. Accuracy

Not applicable.

8. Further Comments

None.

9. Example

The example program prints a set of five pseudo-random variates from a gamma distribution with parameters a = 5.0 and b = 1.0, generated by G05FFF after initialisation by G05CBF.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05FFF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=5)
*     .. Local Scalars ..
      INTEGER          IFAIL, J
*     .. Local Arrays ..
      real             X(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05FFF
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05FFF Example Program Results'
      WRITE (NOUT,*)
      IFAIL = 0
      CALL G05CBF(0)
      WRITE (NOUT,*) 'Gamma Dist --- A=5.0, B=1.0'
*
      CALL G05FFF(5.0e0,1.0e0,N,X,IFAIL)
*
      WRITE (NOUT,99999) (X(J),J=1,N)
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05FFF Example Program Results

Gamma Dist --- A=5.0, B=1.0
    6.7603
    2.9943
    8.3800
    4.5740
    4.9672
```

G05FSF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05FSF generates a vector of pseudo-random variates from a von Mises distribution with concentration parameter κ.

2. Specification

```
SUBROUTINE G05FSF (VK, N, T, IFAIL)
INTEGER        N, IFAIL
real           VK, T(N)
```

3. Description

The von Mises distribution is a symmetric distribution used in the analysis of circular data. The probability density function of this distribution on the circle with mean direction $\mu_0 = 0$ and concentration parameter kappa, κ, can be written as:

$$f(\theta) = \frac{e^{\kappa \cos \theta}}{2\pi I_0(\kappa)},$$

where θ is reduced modulo 2π so that $-\pi \leq \theta < \pi$ and $\kappa \geq 0$. For very small κ the distribution is almost the uniform distribution, whereas for $\kappa \rightarrow \infty$ all the probability is concentrated at one point.

The n variates, $\theta_1, \theta_2, ..., \theta_n$, are generated using an envelope rejection method with a wrapped Cauchy target distribution as proposed by Best and Fisher [1] and described by Dagpunar [2].

4. References

[1] BEST, D.J. and FISHER, N.I.
Efficient simulation of the von Mises distribution.
Appl. Statist., 28, pp. 152-157, 1979.

[2] DAGPUNAR, J.
Principles of Random Number Generation.
Oxford University Press, 1988.

[3] MARDIA, K.V.
Statistics of Directional Data.
Academic Press, London and New York, Ch. 3.4.9, 1972.

5. Parameters

1: VK – ***real***. *Input*

On entry: the concentration parameter, κ, of the required von Mises distribution.

Constraint: VK > 0.0.

2: N – INTEGER. *Input*

On entry: the number of random variates required, n.

Constraint: N $\geq$ 1.

3: T(N) – ***real*** array. *Output*

On exit: the n random variates from the specified von Mises distribution, $\theta_1, \theta_2, ..., \theta_n$.

4: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, VK ≤ 0.0,
or N < 1.

7. Accuracy

Not applicable.

8. Further Comments

For a given number of random variates the generation time increases slightly with increasing κ.

If VK is supplied too large (i.e. VK > SQRT(X02ALF())) then floating point overflow will occur in internal calculation.

9. Example

A set of 4 random variates from a von Mises distribution with $\kappa = 2.0$ are generated and printed.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05FSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=10)
*     .. Local Scalars ..
      INTEGER          I, IFAIL
*     .. Local Arrays ..
      real             X(N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05FSF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05FSF Example Program Results'
      WRITE (NOUT,*)
      IFAIL = 0
*
      CALL G05CBF(0)
*
      WRITE (NOUT,*) 'Von Mises Dist --- VK = 2.0'
*
      CALL G05FSF(2.0e0,N,X,IFAIL)
*
      WRITE (NOUT,99999) (X(I),I=1,N)
      STOP
*
99999 FORMAT (1X,F10.4)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
G05FSF Example Program Results

Von Mises Dist --- VK = 2.0
   -1.6218
   -0.2575
   -0.2038
    0.8379
   -1.0074
   -0.6629
   -0.0986
    0.0252
    0.2702
   -0.5739
```

G05GAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05GAF generates a random orthogonal matrix.

2. Specification

```
SUBROUTINE G05GAF (SIDE, INIT, M, N, A, LDA, WK, IFAIL)
INTEGER         M, N, LDA, IFAIL
real            A(LDA,N), WK(*)
CHARACTER*1     SIDE, INIT
```

3. Description

GO5GAF pre- or post-multiplies an m by n matrix A by a random orthogonal matrix U, overwriting A. The matrix A may optionally be initialized to the identity matrix before multiplying by U, hence U is returned. U is generated using the method of Stewart [1]. The algorithm can be summarized as follows.

Let x_1, x_2,...,x_{n-1} follow independent multinormal distributions with zero means and variances $I\sigma^2$ and dimensions n, n–1,...,2; let $H_j = \text{diag}(I_{j-1}, H_j^*)$, where I_{j-1} is the identity matrix and H_j^* is the Householder transformation that reduces x_j to $r_{jj}e_1$, e_1 being the vector with first element one and the remaining elements zero and r_{jj} being a scalar, and let $D = \text{diag}(\text{sign}(r_{11}),\text{sign}(r_{22}),...,\text{sign}(r_{nn}))$. Then the product $U = DH_1H_2...H_{n-1}$ is a random orthogonal matrix distributed according to the Haar measure over the set of orthogonal matrices of n. See Stewart [1], Theorem 3.3.

4. References

[1] STEWART, G.W.
The efficient generation of random orthogonal matrices with an application to condition estimators.
SIAM J. Numer. Anal. 17, pp. 403-409, 1980.

5. Parameters

1: SIDE – CHARACTER*1. *Input*

On entry: indicates whether the matrix A is multiplied on the left or right by the random orthogonal matrix U.

If SIDE = 'L', the matrix A is multiplied on the left, i.e. pre-multiplied.
If SIDE = 'R', the matrix A is multiplied on the right, i.e. post-multiplied.

Constraint: SIDE = 'L' or 'R'.

2: INIT – CHARACTER*1. *Input*

On entry: indicates whether or not A should be initialised to the identity matrix.

If INIT = 'I' then A is initialised to the identity matrix.
If INIT = 'N' then A is not initialised and the matrix A must be supplied in A.

Constraint: INIT = 'I' or 'N'.

3: M – INTEGER. *Input*

On entry: the number of rows of the matrix A, m.

Constraint: if SIDE = 'L' then M > 1 else M ≥ 1.

4: N – INTEGER. *Input*

On entry: the number of columns of the matrix A, n.

Constraint: if SIDE = 'R' then N > 1 else N ≥ 1.

5: A(LDA,N) – ***real*** array. *Input/Output*

On entry: if INIT = 'N' then A must contain the matrix A.

On exit: the matrix UA when SIDE = 'L' or the matrix AU when SIDE = 'R'.

6: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which G05GAF is called.

Constraint: LDA ≥ M.

7: WK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WK must be at least 2*M if SIDE = 'L' or 2*N if SIDE = 'R'.

8: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, M < 1,
or N < 1,
or LDA < M.

IFAIL = 2

On entry, SIDE ≠ 'L' or 'R',
or INIT ≠ 'I' or 'N'.

IFAIL = 3

On entry, an orthogonal matrix of dimension 1 has been requested.

7. Accuracy

The maximum error in $U^T U$ should be a modest multiple of ***machine precision***.

8. Further Comments

G05GBF computes a random correlation matrix from a random orthogonal matrix.

9. Example

A 4 by 4 orthogonal matrix is generated using the INIT = 'I' option and the result printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05GAF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          N, M, LDA
      PARAMETER        (N=4,M=4,LDA=10)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, J
*     .. Local Arrays ..
      real             A(LDA,N), WK(2*N)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05GAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05GAF Example Program Results'
      WRITE (NOUT,*)
*
      CALL G05CBF(0)
*
      IFAIL = 0
*
      CALL G05GAF('Right','Initialize',M,N,A,LDA,WK,IFAIL)
*
      DO 20 I = 1, M
         WRITE (NOUT,99999) (A(I,J),J=1,N)
   20 CONTINUE
      STOP
*
99999 FORMAT (1X,4F9.3)
      END
```

9.2. Program Data

None.

9.3. Program Results

```
 G05GAF Example Program Results

    -0.461    0.823   -0.251    0.218
     0.446    0.470    0.064   -0.759
    -0.766   -0.204    0.256   -0.554
     0.056    0.245    0.931    0.264
```

G05GBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05GBF generates a random correlation matrix with given eigenvalues.

2. Specification

```
SUBROUTINE G05GBF (N, D, C, LDC, EPS, WK, IFAIL)
INTEGER          N, LDC, IFAIL
real             D(N), C(LDC,N), EPS, WK(2*N)
```

3. Description

Given n eigenvalues, $\lambda_1,\lambda_2,...,\lambda_n$, such that

$$\sum_{i=1}^{n}\lambda_i = n$$

and

$$\lambda_i \geq 0 \text{ for } i = 1,2,...,n,$$

G05GBF will generate a random correlation matrix, C, of dimension n, with eigenvalues $\lambda_1,\lambda_2,...,\lambda_n$.

The method used is based on that described by Lin and Bendel [1]. Let D be the diagonal matrix with values $\lambda_1,\lambda_2,...,\lambda_n$ and let A be a random orthogonal matrix generated by G05GAF then the matrix $C_0 = ADA^T$ is a random covariance matrix with eigenvalues $\lambda_1,\lambda_2,...,\lambda_n$. The matrix C_0 is transformed into a correlation matrix by means of $n-1$ elementary rotation matrices P_i such that $C = P_{n-1}P_{n-2}...P_1C_0P_1^T...P_{n-2}^TP_{n-1}^T$. The restriction on the sum of eigenvalues implies that for any diagonal element of $C_0 > 1$, there is another diagonal element < 1. The P_i are constructed from such pairs, chosen at random, to produce a unit diagonal element corresponding to the first element. This is repeated until all diagonal elements are 1 to within a given tolerance ε.

The randomness of C should be interpreted only to the extent that A is a random orthogonal matrix and C is computed from A using the P_i which are chosen as arbitrarily as possible.

4. References

[1] LIN, S.P. and BENDEL, R.B.
Algorithm AS213. Generation of Population Correlation Matrices with Specified Eigenvalues.
Appl. Statist., 34, pp. 193-198, 1985.

5. Parameters

1: N – INTEGER. *Input*

On entry: the dimension of the correlation matrix to be generated, n.

Constraint: N $\geq$ 1.

2: D(N) – ***real*** array. *Input*

On entry: the n eigenvalues, λ_i, for $i = 1,2,...,n$.

Constraints: D(i) $\geq$ 0.0, for $i = 1,2,...,n$, and $\sum_{i=1}^{n}$D(i) $= n$ to within EPS.

3: C(LDC,N) – ***real*** array. *Output*

On exit: a random correlation matrix, C, of dimension n.

4: LDC – INTEGER. *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which G05GBF is called.

Constraint: LDC $\geq$ N.

5: EPS – ***real.*** *Input*

On entry: the maximum acceptable error in the diagonal elements, ε.

Constraint: EPS $\geq$ N× ***machine precision.***

Suggested value: EPS = 0.00001.

6: WK(2*N) – ***real*** array. *Workspace*

7: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, N < 0,
or LDC < N,
or EPS < N× ***machine precision.***

IFAIL = 2

On entry, D(i) < 0.0 for some i,
or $\sum_{i=1}^{n}$D(i) $\neq n$ to within EPS.

IFAIL = 3

The error in a diagonal element is greater than EPS. The value of EPS should be increased. Otherwise the program could be re-run with a different value used for the seed of the random number generator, see G05CBF or G05CCF.

7. Accuracy

The maximum error in a diagonal element is given by EPS.

8. Further Comments

The time taken by the routine is approximately proportional to n^2.

9. Example

A 3 by 3 correlation matrix with eigenvalues of 0.7, 0.9 and 1.4 is generated and printed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05GBF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=10)
*     .. Local Scalars ..
      real             EPS
      INTEGER          I, IFAIL, J, LDC, N
*     .. Local Arrays ..
      real             C(NMAX,NMAX), D(NMAX), WK(2*NMAX)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05GBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05GBF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) (D(I),I=1,N)
*
         WRITE (NOUT,*)
*
         LDC = NMAX
         CALL G05CBF(0)
         EPS = 0.0001e0
*
         IFAIL = 0
*
         CALL G05GBF(N,D,C,LDC,EPS,WK,IFAIL)
*
         DO 20 I = 1, N
            WRITE (NOUT,99999) (C(I,J),J=1,N)
   20    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,3F9.3)
      END
```

9.2. Program Data

```
G05GBF Example Program Data
3
0.7 0.9 1.4
```

9.3. Program Results

```
 G05GBF Example Program Results

     1.000    0.100   -0.251
     0.100    1.000   -0.239
    -0.251   -0.239    1.000
```

G05HDF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G05HDF generates a realisation of a multivariate time series from a vector autoregressive moving average (VARMA) model. The realisation may be continued or a new realisation generated at subsequent calls to G05HDF.

2. Specification

```
 SUBROUTINE G05HDF (MODE, K, IP, IQ, MEAN, PAR, LPAR, QQ, IK, N, W,
1                   REF, LREF, IWORK, LIWORK, IFAIL)
 INTEGER          K, IP, IQ, LPAR, IK, N, LREF, IWORK(LIWORK), LIWORK,
1                 IFAIL
 real             PAR(LPAR), QQ(IK,K), W(IK,N), REF(LREF)
 CHARACTER*1      MODE, MEAN
```

3. Description

Let the vector $W_t = (w_{1t},w_{2t},\ldots,w_{kt})^T$, denote a k dimensional time series which is assumed to follow a vector autoregressive moving average (VARMA) model of the form:

$$\begin{aligned} W_t-\mu &= \phi_1(W_{t-1}-\mu) + \phi_2(W_{t-2}-\mu) + \ldots + \phi_p(W_{t-p}-\mu) + \\ &\quad \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \ldots - \theta_q\epsilon_{t-q} \qquad (1) \end{aligned}$$

where $\epsilon_t = (\epsilon_{1t},\epsilon_{2t},\ldots,\epsilon_{kt})^T$, is a vector of k residual series assumed to be Normally distributed with zero mean and positive-definite covariance matrix Σ. The components of ϵ_t are assumed to be uncorrelated at non-simultaneous lags. The ϕ_i's and θ_j's are k by k matrices of parameters. $\{\phi_i\}$, for $i = 1,2,\ldots,p$, are called the autoregressive (AR) parameter matrices, and $\{\theta_j\}$, for $j = 1,2,\ldots,q$, the moving average (MA) parameter matrices. The parameters in the model are thus the p k by k ϕ-matrices, the q k by k θ-matrices, the mean vector μ and the residual error covariance matrix Σ. Let

$$A(\phi) = \begin{bmatrix} \phi_1 & I\ 0\ .\ .\ .\ 0 \\ \phi_2 & 0\ I\ 0\ .\ .\ 0 \\ . & \quad . \\ . & \quad\quad . \\ . & \quad\quad . \\ \phi_{p-1} & 0\ .\ .\ .\ 0\ I \\ \phi_p & 0\ .\ .\ .\ 0\ 0 \end{bmatrix}_{pk\times pk} \quad \text{and} \quad B(\theta) = \begin{bmatrix} \theta_1 & I\ 0\ .\ .\ .\ 0 \\ \theta_2 & 0\ I\ 0\ .\ .\ 0 \\ . & \quad . \\ . & \quad\quad . \\ . & \quad\quad . \\ \theta_{q-1} & 0\ .\ .\ .\ 0\ I \\ \theta_q & 0\ .\ .\ .\ 0\ 0 \end{bmatrix}_{qk\times qk}$$

where I denotes the k by k identity matrix.

The model (1) must be both stationary and invertible. The model is said to be stationary if the eigenvalues of $A(\phi)$ lie inside the unit circle and invertible if the eigenvalues of $B(\theta)$ lie inside the unit circle.

For $k \geq 6$ the VARMA model (1) is recast into state space form and a realisation of the state vector at time zero computed. For all other cases the routine computes a realisation of the pre-observed vectors $W_0,W_{-1},\ldots,W_{1-p}$, $\epsilon_0,\epsilon_{-1},\ldots,\epsilon_{1-q}$, from equation (1), see Shea [2]. This realisation is then used to generate a sequence of successive time series observations. Note that special action is taken for pure MA models, that is for $p = 0$.

At the user's request a new realisation of the time series may be generated with less computation using only the information saved in a reference vector from a previous call to G05HDF. See the description of the parameter MODE in Section 5 for details.

The routine returns a realisation of $W_1,W_2,...,W_n$. On a successful exit, the recent history is updated and saved in the array REF so that G05HDF may be called again to generate a realisation of $W_{n+1},W_{n+2},...,$etc. See the description of the parameter MODE in Section 5 for details.

Further computational details are given in [2]. Note however that this routine uses a spectral decomposition rather than a Cholesky factorisation to generate the multivariate Normals. Although this method involves more multiplications than the Cholesky factorisation method and is thus slightly slower it is more stable when faced with ill-conditioned covariance matrices. A method of assigning the AR and MA coefficient matrices so that the stationarity and invertibility conditions are satisfied is described in Barone [1].

4. References

[1] BARONE, P.
A Method for Generating Independent Realisations of a Multivariate Normal Stationary and Invertible ARMA(p,q) Process.
J. Time Series Anal., 8, pp. 125-130, 1987.

[2] SHEA, B.L.
A Note on the Generation of Independent Realisations of a Vector Autoregressive Moving Average Process.
J. Time Series Anal., 9, pp. 403-410, 1988.

5. Parameters

1: MODE – CHARACTER*1. *Input*

On entry: must be set as follows;

MODE = 'S' (Start), the routine is being called for the first time; a realisation of the recent history is computed, and the sequence of time series values from the VARMA model is then generated.

MODE = 'R' (Restart), the routine must have been called before with the same VARMA model; a new realisation of the recent history is computed using information stored in the reference vector, followed by the sequence of time series values.

MODE = 'C' (Continue), the routine must have been called before with the same VARMA model; a new sequence is generated, from the point at which the last sequence ended, using a realisation of the recent history which was updated and stored by the previous call to the routine.

If MODE = 'R' or 'C', then the user must ensure that the reference vector REF and the values of K, IP, IQ, MEAN, PAR, QQ and IK have not been changed between calls to G05HDF.

Constraint: MODE = 'S', 'R', or 'C'.

2: K – INTEGER. *Input*

On entry: the dimension k, of the multivariate time series.

Constraint: K $\geq$ 1.

3: IP – INTEGER. *Input*

On entry: the number of AR parameter matrices, p.

Constraint: IP $\geq$ 0.

4: IQ – INTEGER. *Input*

On entry: the number of MA parameter matrices, q.

Constraint: IQ $\geq$ 0.

5: MEAN – CHARACTER*1. *Input*

On entry: indicates whether or not all elements of μ are to be supplied by the user or to be taken as zero.

If MEAN = 'M' the values of μ, are supplied in the array PAR.
If MEAN = 'Z', all elements of μ are to be taken as zero.

Constraint: MEAN = 'M' or 'Z'.

6: PAR(LPAR) – ***real*** array. *Input*

On entry: the parameter values read in row by row in the order $\phi_1,\phi_2,...,\phi_p$, $\theta_1,\theta_2,...,\theta_q$, μ.

Thus, if IP > 0, then PAR($(l-1)\times k\times k+(i-1)\times k+j$) must be set equal to the (i,j)th element of ϕ_l, for $l = 1,2,...,p$; $i,j = 1,2,...,k$.

If IQ > 0, then PAR($p\times k\times k+(l-1)\times k\times k+(i-1)\times k+j$) must be set equal to the (i,j)th element of θ_l, for $l = 1,2,...,q$; $i,j = 1,2,...,k$.

If MEAN = 'M', then PAR($(p+q)\times k\times k+i$) must be set equal to the ith component of the mean vector μ, for $i = 1,2,...,k$.

Constraint: the first IP×K×K elements of PAR must satisfy the stationarity condition and the next IQ×K×K elements of PAR must satisfy the invertibility condition.

7: LPAR – INTEGER. *Input*

On entry: the dimension of the array PAR as declared in the (sub)program from which G05HDF is called.

Constraints: LPAR ≥ max(1,npar) where
npar = (IP+IQ)×K×K if MEAN = 'Z' and
npar = (IP+IQ)×K×K + K if MEAN = 'M'.

8: QQ(IK,K) – ***real*** array. *Input*

On entry: QQ(i,j) must contain the (i,j)th element of Σ. Only the lower triangle is required.

Constraint: the elements of QQ must be such that Σ is positive-definite.

9: IK – INTEGER. *Input*

On entry: the first dimension of the arrays QQ and W as declared in the (sub)program from which G05HDF is called.

Constraint: IK ≥ K.

10: N – INTEGER. *Input*

On entry: the number of observations to be generated, n.

Constraint: N ≥ 1.

11: W(IK,N) – ***real*** array. *Output*

On exit: W(i,t) will contain a realisation of the ith component of W_t, for $i = 1,2,...,k$; $t = 1,2,...,n$.

12: REF(LREF) – ***real*** array. *Input/Output*

On entry: if MODE = 'R' or 'C', then the array REF as output from the previous call to G05HDF must be input without any change to the first $m + (k+1)(k+2) + (m+1)(m+2)$ elements where $m = k\times\max(p,q)$ if $k \geq 6$ and $k(p+q)$ if $k < 6$.

If MODE = 'S', then the contents of REF need not be set.

On exit: the first $m + (k+1)(k+2) + (m+1)(m+2)$ elements of the array REF contain information required for any subsequent calls to the routine with MODE = 'R' or 'C'; the rest of the array is used as workspace. See Section 8.

13: LREF – INTEGER. *Input*

On entry: the dimension of the array REF as declared in the (sub)program from which G05HDF is called.

Constraint:

Let r = max(IP,IQ)

and l = K(K+1)/2 if IP = 0,
K(K+1)/2 + (IP−1)K^2 if IP ≥ 1,

If K ≥ 6, then LREF ≥ $(5r^2+1)\mathrm{K}^2+(4r+3)\mathrm{K}+4$

If K < 6, then LREF ≥ $((\mathrm{IP}+\mathrm{IQ})^2+1)\mathrm{K}^2+(4(\mathrm{IP}+\mathrm{IQ})+3)\mathrm{K}$ + $\max\{\mathrm{K}r(\mathrm{K}r+2),\ \mathrm{K}^2(\mathrm{IP}+\mathrm{IQ})^2+l(l+3)+\mathrm{K}^2(\mathrm{IQ}+1)\}$ + 4.

See Section 8 for some examples of the required size of the array REF.

14: IWORK(LIWORK) – INTEGER array. *Workspace*

15: LIWORK – INTEGER. *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which G05HDF is called.

Constraint: LIWORK ≥ K×max(IP,IQ).

16: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry,	MODE ≠ 'S', 'R' or 'C',
or	K < 1,
or	IP < 0,
or	IQ < 0,
or	MEAN ≠ 'M', or 'Z',
or	LPAR < max((IP+IQ)×K×K+K,1) and MEAN = 'M',
or	LPAR < max((IP+IQ)×K×K,1) and MEAN = 'Z',
or	IK < K,
or	N < 1,
or	LREF is too small,
or	LIWORK is too small.

IFAIL = 2

On entry, either the value of Σ is not positive-definite, or the AR parameters are such that the model is non-stationary, or the MA parameters are such that the model is non-invertible. To proceed, the user must try different parameter values.

IFAIL = 3

This is an unlikely exit brought about by an excessive number of iterations being needed by the NAG Fortran Library routine used to evaluate the eigenvalues of $A(\phi)$ or $B(\theta)$. If this error occurs please contact NAG.

IFAIL = 4

G05HDF has not been able to calculate all the required elements of the array REF. This is an unlikely exit brought about by an excessive number of iterations being needed by F02ABF to evaluate eigenvalues to be stored in the array REF. If this error occurs please contact NAG.

IFAIL = 5

G05HDF has not been able to calculate all the required elements of the array REF. This is likely to be because the AR parameters are very close to the boundary of the stationarity region.

IFAIL = 6

The reference vector REF has been corrupted, when MODE is set to 'R' or 'C'. To proceed, the user should set MODE to 'S'.

7. Accuracy

The accuracy is limited by the matrix computations performed, and this is dependent on the condition of the parameter and covariance matrices.

8. Further Comments

Note that, in reference to IFAIL = 2, G05HDF will permit MA parameters on the boundary of the invertibility region.

The elements of REF contain amongst other information details of the spectral decompositions which are used to generate future multivariate Normals. Note that these eigenvectors may not be unique on different machines. For example the eigenvectors corresponding to multiple eigenvalues may be permuted. Although an effort is made to ensure that the eigenvectors have the same sign on all machines, differences in the signs may theoretically still occur.

The following table gives some examples of the required size of the array REF, specified by the parameter LREF, for $k = 1,2,3$, and for various values of p and q.

			q		
		0	1	2	3
		13	20	31	46
	0	36	56	92	144
		85	124	199	310
		19	30	45	64
	1	52	88	140	208
		115	190	301	448
p					
		35	50	69	92
	2	136	188	256	340
		397	508	655	838
		57	76	99	126
	3	268	336	420	520
		877	1024	1207	1426

Note that the routine G13DXF may be used to check whether a VARMA model is stationary and invertible.

The time taken depends on the values of p, q and especially n and k.

9. Example

This program generates two realisations, each of length 48, from the bivariate AR(1) model

$$W_t - \mu = \phi_1(W_{t-1}-\mu) + \varepsilon_t$$

with

$$\phi_1 = \begin{bmatrix} 0.80 & 0.07 \\ 0.00 & 0.58 \end{bmatrix}, \ \mu = \begin{bmatrix} 5.00 \\ 9.00 \end{bmatrix}, \text{ and } \Sigma = \begin{bmatrix} 2.97 & \\ 0.64 & 5.38 \end{bmatrix}.$$

In the first call MODE is set to 'S' in order to set up the reference vector before generating the first realisation. In the subsequent call MODE is set to 'R' and a new recent history is generated and used to generate the second realisation.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G05HDF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          KMAX, IK, IPMAX, IQMAX, LPARMX, NMAX, LREF,
     +                 LIWORK
      PARAMETER        (KMAX=3,IK=KMAX,IPMAX=2,IQMAX=2,
     +                 LPARMX=(IPMAX+IQMAX)*KMAX*KMAX+KMAX,NMAX=100,
     +                 LREF=554,LIWORK=10)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IP, IQ, J, K, N, NPAR
      CHARACTER        MEAN
*     .. Local Arrays ..
      real             PAR(LPARMX), QQ(IK,KMAX), REF(LREF), W(IK,NMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         G05CBF, G05HDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G05HDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) K, IP, IQ, N, MEAN
*
      IF (K.GT.0 .AND. K.LE.KMAX .AND. IP.GE.0 .AND. IP.LE.IPMAX .AND.
     +    IQ.GE.0 .AND. IQ.LE.IQMAX) THEN
         NPAR = (IP+IQ)*K*K
         IF (MEAN.EQ.'M' .OR. MEAN.EQ.'m') NPAR = NPAR + K
         IF (N.GT.0 .AND. N.LE.NMAX) THEN
            READ (NIN,*) (PAR(I),I=1,NPAR)
            DO 20 I = 1, K
               READ (NIN,*) (QQ(I,J),J=1,I)
   20       CONTINUE
*
            CALL G05CBF(0)
*
            IFAIL = 0
*
            CALL G05HDF('Start',K,IP,IQ,MEAN,PAR,NPAR,QQ,IK,N,W,REF,
     +                  LREF,IWORK,LIWORK,IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Realisation Number 1'
*
            DO 40 I = 1, K
               WRITE (NOUT,99999) ' Series number ', I
               WRITE (NOUT,*) ' -------------'
               WRITE (NOUT,*)
               WRITE (NOUT,99998) (W(I,J),J=1,N)
   40       CONTINUE
```

```
*
            IFAIL = 0
*
            CALL G05HDF('Restart',K,IP,IQ,MEAN,PAR,NPAR,QQ,IK,N,W,REF,
     +                   LREF,IWORK,LIWORK,IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Realisation Number 2'
*
            DO 60 I = 1, K
               WRITE (NOUT,99999) '  Series number ', I
               WRITE (NOUT,*) '  -------------'
               WRITE (NOUT,*)
               WRITE (NOUT,99998) (W(I,J),J=1,N)
   60       CONTINUE
*
         END IF
      END IF
      STOP
*
99999 FORMAT (/1X,A,I3)
99998 FORMAT (8(2X,F8.3))
      END
```

9.2. Program Data

```
G05HDF Example Program Data
 2 1 0 48 'M'                      : K, IP, IQ, N, MEAN
 0.80 0.07 0.00 0.58 5.00 9.00     : PAR
 2.97                              : QQ
 0.64 5.38
```

9.3. Program Results

```
 G05HDF Example Program Results

  Realisation Number 1

   Series number   1
   -------------

     4.722     6.101     3.707     2.501     2.757     7.143     4.752     5.624
     5.197     3.596     4.752     4.441     4.733     4.970     5.585     4.820
     4.847     1.414     0.548     1.212     0.203    -1.066    -1.992    -1.765
    -0.948    -0.311     5.809     2.649     6.345     3.522     3.982     2.394
    -0.972    -1.839    -2.293    -1.304    -2.571    -0.447     2.301    -1.910
    -2.155    -0.375     1.737     3.194     2.236     3.504     4.163     5.562

   Series number   2
   -------------

     1.434    -0.767     0.403    -3.162     1.674     1.241     0.596    -2.187
    -0.681    -3.079    -0.786     4.618     3.477     2.454     4.775     0.451
     3.902    -0.017    -3.620    -1.489    -4.478    -5.614    -5.265     0.275
    -1.325    -5.113    -1.932    -1.989    -2.075    -3.710    -3.205    -5.205
    -4.857    -0.731     1.350    -2.720    -0.110    -0.161     1.944     2.219
     0.387     2.266     2.049     0.214     0.638    -0.026    -0.822     1.735


  Realisation Number 2

   Series number   1
   -------------

     1.289     2.151     0.168     2.621     3.190     0.488     1.254    -0.500
    -2.296    -4.731    -4.677    -2.975    -0.964     0.694    -1.225    -0.809
    -3.127    -1.230    -2.990    -5.861    -3.854    -6.995    -5.921    -4.316
    -5.010    -6.696    -6.756    -5.418    -5.828    -4.878    -4.853    -4.411
    -2.006    -1.415    -2.847    -3.532    -3.479    -0.209     0.034     0.565
     0.962     2.168     2.168     3.531     2.702     1.524    -2.152     0.718
```

```
Series number   2
-------------

 -0.521    -2.962    -3.000    -0.633    -2.936    -5.076    -2.939    -1.469
 -3.834    -1.155    -4.113    -3.726    -0.830    -0.403    -4.221    -3.672
  2.775     0.979    -0.732    -0.063     1.502    -3.152    -6.403    -5.306
 -3.899    -5.066    -2.451    -0.843    -1.178    -4.910    -5.041    -1.291
  0.818     1.957    -2.845     2.020     0.847    -2.144    -1.480    -1.286
  1.859    -6.834    -3.248    -1.032    -1.977    -1.491    -0.719    -0.196
```

Chapter G07 – Univariate Estimation

Note. Please refer to the Users' Note for your implementation to check that a routine is available.

Routine Name	Mark of Introduction	Purpose
G07AAF	15	Computes confidence interval for the parameter of a binomial distribution
G07ABF	15	Computes confidence interval for the parameter of a Poisson distribution
G07BBF	15	Computes maximum likelihood estimates for parameters of the Normal distribution from grouped and/or censored data
G07BEF	15	Computes maximum likelihood estimates for parameters of the Weibull distribution
G07CAF	15	Computes t-test statistic for a difference in means between two Normal populations, confidence interval
G07DAF	13	Robust estimation, median, median absolute deviation, robust standard deviation
G07DBF	13	Robust estimation, M-estimates for location and scale parameters, standard weight functions
G07DCF	13	Robust estimation, M-estimates for location and scale parameters, user-defined weight functions
G07DDF	14	Computes a trimmed and winsorized mean of a single sample with estimates of their variance
G07EAF	16	Robust confidence intervals, 1 sample
G07EBF	16	Robust confidence intervals, 2 sample

Chapter G07

Univariate Estimation

Contents

1 Scope of the Chapter

This chapter deals with the estimation of unknown parameters of a univariate distribution. It includes both point and interval estimation using maximum likelihood and robust methods.

2 Background to the Problems

Statistical inference is concerned with the making of inferences about a **population** using the observed part of the population called a **sample**. The population can usually be described using a probability model which will be written in terms of some unknown **parameters**. For example, the hours of relief given by a drug may be assumed to follow a Normal distribution with mean μ and variance σ^2; it is then required to make inferences about the parameters, μ and σ^2, on the basis of an observed sample of relief times.

There are two main aspects of statistical inference: the **estimation** of the parameters and the **testing of hypotheses** about the parameters. In the example above, the values of the parameter σ^2 may be estimated and the hypothesis that $\mu \geq 3$ tested. This chapter is mainly concerned with estimation but the test of a hypothesis about a parameter is often closely linked to its estimation. Tests of hypotheses which are not linked closely to estimation are given in the chapter on non-parametric statistics (Chapter G08).

There are two types of estimation to be considered in this chapter: **point estimation** and **interval estimation**. Point estimation is when a single value is obtained as the best estimate of the parameter. However, as this estimate will be based on only one of a large number of possible samples, it can be seen that if a different sample were taken, a different estimate would be obtained. The distribution of the estimate across all the possible samples is known as the **sampling distribution**. The sampling distribution contains information on the performance of the estimator, and enables estimators to be compared. For example, a good estimator would have a sampling distribution with mean equal to the true value of the parameter; that is, it should be an **unbiased** estimator; also the variance of the sampling distribution should be as small as possible. When considering a parameter estimate it is important to consider its variability as measured by its variance, or more often the square root of the variance, the **standard error**.

The sampling distribution can be used to find interval estimates or confidence intervals for the parameter. A **confidence interval** is an interval calculated from the sample so that its distribution, as given by the sampling distribution, is such that it contains the true value of the parameter with a certain probability.

Estimates will be functions of the observed sample and these functions are known as **estimators**. It is usually more convenient for the estimator to be based on statistics from the sample rather than all the individuals observations. If these statistics contain all the relevant information then they are known as **sufficient statistics**. There are several ways of obtaining the estimators; these include least-squares, the method of moments, and **maximum likelihood**. Least-squares estimation requires no knowledge of the distributional form of the error apart from its mean and variance matrix, whereas the method of maximum likelihood is mainly applicable to situations in which the true distribution is known apart from the values of a finite number of unknown parameters. Note that under the assumption of Normality, the least-squares estimation is equivalent to the maximum likelihood estimation. Least squares is often used in regression analysis as described in Chapter G02, and maximum likelihood is described below.

Estimators derived from least-squares or maximum likelihood will often be greatly affected by the presence of extreme or unusual observations. Estimators that are designed to be less affected are known as **robust estimators**.

2.1 Maximum Likelihood Estimation

Let X_i be a univariate random variable with probability density function

$$f_{X_i}(x_i;\theta),$$

where θ is a vector of length p consisting of the unknown parameters. For example, a Normal distribution with mean θ_1 and standard deviation θ_2 has probability density function

$$\frac{1}{\sqrt{2\pi}\theta_2}\exp\left(-\frac{1}{2}\left(\frac{x_i-\theta_1}{\theta_2}\right)^2\right).$$

The likelihood for a sample of n independent observations is

$$\text{Like} = \prod_{i=1}^{n} f_{X_i}(x_i;\theta),$$

where x_i is the observed value of X_i. If each X_i has an identical distribution, this reduces to

$$\text{Like} = \prod_{i=1}^{n} f_X(x_i;\theta), \tag{1}$$

and the log-likelihood is

$$\log(\text{Like}) = L = \sum_{i=1}^{n} \log(f_X(x_i;\theta)). \tag{2}$$

The maximum likelihood estimates $(\hat{\theta})$ of θ are the values of θ that maximize (1) and (2). If the range of X is independent of the parameters, then $\hat{\theta}$ can usually be found as the solution to

$$\sum_{i=1}^{n} \frac{\partial}{\partial\hat{\theta}_j} \log(f_X(x_i;\hat{\theta})) = \frac{\partial L}{\partial\hat{\theta}_j} = 0, \quad j = 1,2,\ldots,p. \tag{3}$$

Note that $\frac{\partial L}{\partial\theta_j}$ is known as the efficient score.

Maximum likelihood estimators possess several important properties.

(a) Maximum likelihood estimators are functions of the sufficient statistics.

(b) Maximum likelihood estimators are (under certain conditions) **consistent**. That is, the estimator converges in probability to the true value as the sample size increases. Note that for small samples the maximum likelihood estimator may be biased.

(c) For maximum likelihood estimators found as a solution to (3), subject to certain conditions, it follows that

$$E\left(\frac{\partial L}{\partial\theta}\right) = 0, \tag{4}$$

and

$$I(\theta) = -E\left(\frac{\partial^2 L}{\partial\theta^2}\right) = E\left(\left(\frac{\partial L}{\partial\theta}\right)^2\right), \tag{5}$$

and then that $\hat{\theta}$ is asymptotically Normal with mean vector θ_0 and variance-covariance matrix $I_{\theta_0}^{-1}$ where θ_0 denotes the true value of θ. The matrix I_θ is known as the information matrix and $I_{\theta_0}^{-1}$ is known as the Cramer–Rao lower bound for the variance of an estimator of θ.

For example, if we consider a sample, $x_1, x_2, \ldots, x_n$, of size n drawn from a Normal distribution with unknown mean μ and unknown variance σ_2 then we have

$$L = \log(\text{Like}(\mu,\sigma^2;x)) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \sum_{i=1}^{n}(x_i-\mu)^2/2\sigma^2$$

and thus

$$\frac{\partial L}{\partial\mu} = \sum_{i=1}^{n}(x_i-\mu)/\sigma^2$$

and

$$\frac{\partial L}{\partial\sigma^2} = -\frac{n}{2\sigma^2} + \sum_{i=1}^{n}(x_i-\mu)^2/2\sigma^4.$$

Then equating these two equations to zero and solving gives the maximum likelihood estimates

$$\hat{\mu} = \bar{x}$$

and

$$\hat{\sigma}^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2/n.$$

These maximum likelihood estimates are asymptotically Normal with mean vector a, where

$$a^T = (\mu, \sigma^2),$$

and covariance matrix C. To obtain C we find the second derivatives of L with respect to μ and σ^2 as follows:

$$\frac{\partial^2 L}{\partial \mu^2} = -\frac{n}{\sigma^2}$$

$$\frac{\partial^2 L}{\partial (\sigma^2)^2} = \frac{n}{2\sigma^4} - \sum_{i=1}^{n}(x_i - \mu)^2/\sigma^6$$

$$\frac{\partial^2 L}{\partial \mu \partial \sigma^2} = \frac{\partial^2 L}{\partial \sigma^2 \partial \mu} = -\frac{n(\bar{x} - \mu)}{\sigma^4}.$$

Then

$$C^{-1} = -E\begin{pmatrix} \dfrac{\partial^2 L}{\partial \mu^2} & \dfrac{\partial^2 L}{\partial \sigma^2 \partial \mu} \\ \dfrac{\partial^2 L}{\partial \mu \partial \sigma^2} & \dfrac{\partial^2 L}{\partial (\sigma^2)^2} \end{pmatrix} = \begin{pmatrix} n/\sigma^2 & 0 \\ 0 & n/2\sigma^4 \end{pmatrix}$$

so that

$$C = \begin{pmatrix} \sigma^2/n & 0 \\ 0 & 2\sigma^4/n \end{pmatrix}.$$

To obtain an estimate of C the matrix may be evaluated at the maximum likelihood estimates.

It may not always be possible to find maximum likelihood estimates in a convenient closed form, and in these cases iterative numerical methods, such as the Newton–Raphson procedure or the EM algorithm (expectation maximisation), will be neccessary to compute the maximum likelihood estimates. Their asymptotic variances and covariances may then be found by substituting the estimates into the second derivatives. Note that it may be difficult to find the expected value of the second derivatives required for the variance-covariance matrix and in these cases the observed value of the second derivatives is often used.

The use of maximum likelihood estimation allows the construction of generalized likelihood ratio tests. If $\lambda = 2(l_1 - l_2)$ where l_1 is the maximised log-likelihood function for a model 1 and l_2 is the maximised log-likelihood function for a model 2, then under the hypothesis that model 2 is correct, 2λ is asymptotically distributed as a χ^2 variable with $p - q$ degrees of freedom. Consider two models in which model 1 has p parameters and model 2 is a sub-model (nested model) of model 1 with $q < p$ paramters, that is model 1 has an extra $p - q$ parameters. This result provides a useful method for performing hypothesis tests on the parameters. Alternatively, tests exist based on the asymptotic Normality of the estimator and the efficient score; see Cox and Hinkley [1], page 315.

2.2 Confidence Intervals

Suppose we can find a function, $t(x, \theta)$, whose distribution depends upon the sample x but not on the unknown parameter θ, and which is a monotonic (say decreasing) function in θ for each x, then we can find t_1 such that $P(t_1 \leq t(x, \theta)) = 1 - \alpha$ no matter what θ happens to be. The function $t(x, \theta)$ is known as a pivotal quantity. Since the function is monotonic the statement that $t_1 \leq t(x, \theta)$ may be rewritten as $\theta \geq \theta_1(x)$, see Figure 1. The statistic $\theta_1(x)$ will vary from sample to sample and if we assert that $\theta \geq \theta_1(x)$ for any sample values which arise, we will be right in a proportion $1 - \alpha$ of the cases, in the long run or on average. We call $\theta_1(x)$ a $1 - \alpha$ upper confidence limit for θ.

We have considered only an upper confidence limit. The above idea may be generalised to a two-sided confidence interval where two quantities, t_0 and t_1, are found such that for all θ, $\mathrm{P}(t_1 \leq t(x, \theta) \leq t_0) = 1 - \alpha$. This interval may be rewritten as $\theta_0(x) \leq \theta \leq \theta_1(x)$. Thus if we assert that θ lies in the interval $[\theta_0(x), \theta_1(x)]$ we will be right on average in $1 - \alpha$ proportion of the times under repeated sampling.

Hypothesis (significance) tests on the parameters may be used to find these confidence limits. For example, if we observe a value, k, from a binomial distribution, with known parameter n and unknown parameter

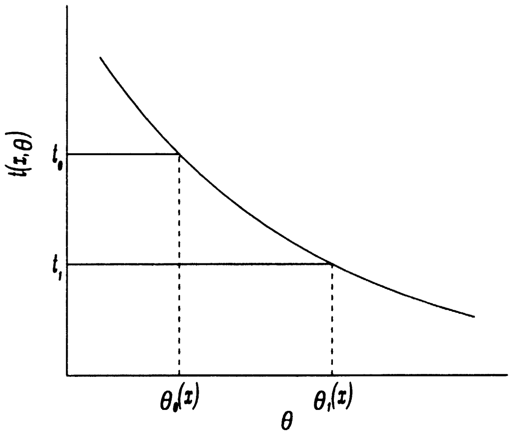


Figure 1

p, then to find the lower confidence limit we find p_l such that the probability that the null hypothesis H_0: $p = p_l$ (against the one sided alternative that $p > p_l$) will be rejected, is less than or equal to $\alpha/2$. Thus for a binomial random variable, B, with parameters n and p_l we require that $P(B \geq k) \leq \alpha/2$. The upper confidence limit, p_u, can be constructed in a similar way.

For large samples the asymptotic Normality of the maximum likelihood estimates discussed above is used to construct confidence intervals for the unknown parameters.

2.3 Robust Estimation

For particular cases the probability density function can be written as

$$f_{X_i}(x_i;\theta) = \frac{1}{\theta_2} g\left(\frac{x_i - \theta_1}{\theta_2}\right),$$

for a suitable function g; then θ_1 is known as a location parameter and θ_2, usually written as σ, is known as a scale parameter. This is true of the Normal distribution.

If θ_1 is a location parameter, as described above, then equation (3) becomes

$$\sum_{i=1}^{n} \psi\left(\frac{x_i - \hat{\theta}_1}{\hat{\sigma}}\right) = 0, \tag{6}$$

where $\psi(z) = -\frac{d}{dz}\log(g(z))$.

For the scale parameter σ (or σ^2) the equation is

$$\sum_{i=1}^{n} \chi\left(\frac{x_i - \hat{\theta}_1}{\hat{\sigma}}\right) = n/2, \tag{7}$$

where $\chi(z) = z\psi(z)/2$.

For the Normal distribution $\psi(z) = z$ and $\chi(z) = z^2/2$. Thus, the maximum likelihood estimates for θ_1 and σ^2 are the sample mean and variance with the n divisor respectively. As the latter is biased, (7) can be replaced by

$$\sum_{i=1}^{n} \chi\left(\frac{x_i - \hat{\theta}_1}{\hat{\sigma}}\right) = (n-1)\beta, \tag{8}$$

where β is a suitable constant, which for the Normal χ function is $\frac{1}{2}$.

The influence of an observation on the estimates depends on the form of the ψ and χ functions. For a discussion of influence, see Hampel *et al.* [2] and Huber [3]. The influence of extreme values can be reduced by bounding the values of the ψ- and χ-functions. One suggestion due to Huber [3] is

$$\psi(z) = \begin{cases} -C, & z < -C \\ z, & |z| \leq C \\ C, & z > C. \end{cases}$$

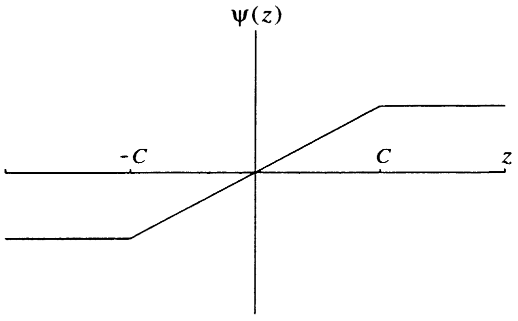


Figure 2

Redescending ψ-functions are often considered; these give zero values to $\psi(z)$ for large positive or negative values of z. Hampel [2] suggested

$$\psi(z) = \begin{cases} -\psi(-z) & \\ z, & 0 \leq z \leq h_1 \\ h_1, & h_1 \leq z \leq h_2 \\ h_1(h_3 - z)/(h_3 - h_2), & h_2 \leq z \leq h_3 \\ 0, & z > h_3. \end{cases}$$

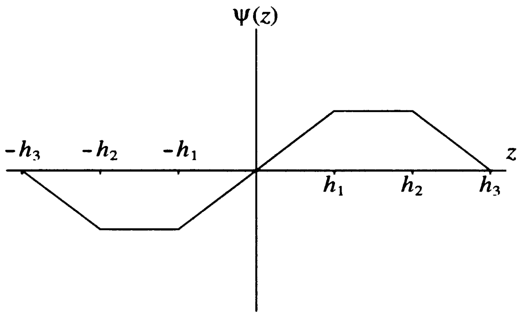


Figure 3

Usually a χ-function based on Huber's ψ-function is used: $\chi = \psi^2/2$. Estimators based on such bounded ψ-functions are known as M-estimators, and provide one type of robust estimator.

Other robust estimators for the location parameter are:

(i) the sample median,

(ii) the trimmed mean, i.e., the mean calculated after the extreme values have been removed from the sample,

(iii) the winsorized mean, i.e., the mean calculated after the extreme values of the sample have been replaced by other more moderate values from the sample.

For the scale parameter, alternative estimators are:

(i) the median absolute deviation scaled to produce an estimator which is unbiased in the case of data coming from a Normal distribution,

(ii) the winsorized variance, i.e., the variance calculated after the extreme values of the sample have been replaced by other more moderate values from the sample.

For a general discussion of robust estimation, see Hampel *et al.* [2] and Huber [3].

2.4 Robust Confidence Intervals

In Section 2.2 it was shown how tests of hypotheses can be used to find confidence intervals. That approach uses a parametric test that requires the assumption that the data used in the computation of the confidence has a known distribution. As an alternative, a more robust confidence interval can be found by replacing the parametric test by a non-parametric test. In the case of the confidence interval for the location parameter, a Wilcoxon test statistic can be used, and for the difference in location, computed from two samples, a Mann–Whitney test statistic can be used.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

Maximum likelihood estimation and confidence intervals.

G07AAF	provides a confidence interval for the parameter p of the binomial distribution.
G07ABF	provides a confidence interval for the mean parameter of the Poisson distribution.
G07BBF	provides maximum likelihood estimates and their standard errors for the parameters of the Normal distribution from grouped and/or censored data.
G07BEF	provides maximum likelihood estimates and their standard errors for the parameters of the Weibull distribution from data which may be right-censored.
G07CAF	provides a t-test statistic to test for a difference in means between two Normal populations, together with a confidence interval for the difference between the means.

Robust estimation.

G07DBF	provides M-estimates for location and, optionally, scale using four common forms of the ψ-function.
G07DCF	produces the M-estimates for location and, optionally, scale but for user-supplied ψ- and χ-functions.
G07DAF	provides the sample median, median absolute deviation, and the scaled value of the median absolute deviation.
G07DDF	provides the trimmed mean and winsorized mean together with estimates of their variance based on a winsorized variance.

Robust Internal Estimation.

G07EAF	produces a rank based confidence interval for locations.
G07EBF	produces a rank based confidence interval for the difference in location between two populations.

4 References

[1] Cox D R and Hinkley D V (1974) *Theoretical Statistics* Chapman and Hall

[2] Hampel F R, Ronchetti E M, Rousseeuw P J and Stahel W A (1986) *Robust Statistics. The Approach Based on Influence Functions* Wiley

[3] Huber P J (1981) *Robust Statistics* Wiley

[4] Kendall M G and Stuart A (1973) *The Advanced Theory of Statistics (Volume 2)* Griffin (3rd Edition)

[5] Silvey S D (1975) *Statistical Inference* Chapman and Hall

G07AAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07AAF computes a confidence interval for the parameter p (the probability of a success) of a binomial distribution.

2. Specification

```
SUBROUTINE G07AAF (N, K, CLEVEL, PL, PU, IFAIL)
INTEGER          N, K, IFAIL
real             CLEVEL, PL, PU
```

3. Description

Given the number of trials, n, and the number of successes, k, this routine computes a $100(1-\alpha)\%$ confidence interval for p, the probability parameter of a binomial distribution with probability function,

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x} \quad \text{for } x = 0,1,\ldots,n,$$

where α is in the interval $(0,1)$.

Let the confidence interval be denoted by $[p_l, p_u]$.

The point estimate for p is $\hat{p} = k/n$.

The lower and upper confidence limits p_l and p_u are estimated by the solutions to the equations;

$$\sum_{x=k}^{n} \binom{n}{x} p_l^x (1-p_l)^{n-x} = \alpha/2,$$

$$\sum_{x=0}^{k} \binom{n}{x} p_u^x (1-p_u)^{n-x} = \alpha/2.$$

Three different methods are used depending on the number of trials, n, and the number of successes, k.

(1) If $\max(k,n-k) < 10^6$.

The relationship between the beta and binomial distributions (see Hastings and Peacock [1], page 38) is used to derive the equivalent equations,

$$p_l = \beta_{k,n-k+1,\alpha/2},$$

$$p_u = \beta_{k+1,n-k,1-\alpha/2},$$

where $\beta_{a,b;\delta}$ is the deviate associated with the lower tail probability, δ, of the beta distribution with parameters a and b. These beta deviates are computed using G01FEF.

(2) If $\max(k,n-k) \geq 10^6$ and $\min(k,n-k) \leq 1000$.

The binomial variate with parameters n and p is approximated by a Poisson variate with mean np, see [1], page 38.
The relationship between the Poisson and χ^2 distributions (see [1], page 112) is used to derive the following equations;

$$p_l = \frac{1}{2n}\chi^2_{2k,\alpha/2},$$

$$p_u = \frac{1}{2n}\chi^2_{2k+2,1-\alpha/2},$$

where $\chi^2_{\delta,\nu}$ is the deviate associated with the lower tail probability, δ, of the χ^2 distribution with ν degrees of freedom.

In turn the relationship between the χ^2 distribution and the gamma distribution (see [1], page 70) yields the following equivalent equations;

$$p_l = \frac{1}{2n}\gamma_{k,2;\alpha/2},$$

$$p_u = \frac{1}{2n}\gamma_{k+1,2;1-\alpha/2},$$

where $\gamma_{\alpha,\beta;\delta}$ is the deviate associated with the lower tail probability, δ, of the gamma distribution with shape parameter α and scale parameter β. These deviates are computed using G01FFF.

(3) If $\max(k,n-k) > 10^6$ and $\min(k,n-k) > 1000$.

The binomial variate with parameters n and p is approximated by a Normal variate with mean np and variance $np(1-p)$, see [1], page 38.
The approximate lower and upper confidence limits p_l and p_u are the solutions to the equations;

$$\frac{k - np_l}{\sqrt{np_l(1-p_l)}} = z_{1-\alpha/2},$$

$$\frac{k - np_u}{\sqrt{np_u(1-p_u)}} = z_{\alpha/2},$$

where z_δ is the deviate associated with the lower tail probability, δ, of the standard Normal distribution. These equations are solved using C02AJF.

4. References

[1] HASTINGS, N.A.J. and PEACOCK, J.B.
Statistical Distributions.
Butterworth, 1975.

[2] SNEDECOR, G.W. and COCHRAN, W.G.
Statistical Methods.
Iowa State University Press, 1967.

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of trials, n.

Constraint: N $\geq$ 1.

2: K – INTEGER. *Input*

On entry: the number of successes, k.

Constraint: 0 $\leq$ K $\leq$ N.

3: CLEVEL – ***real***. *Input*

On entry: the confidence level, $(1-\alpha)$, for two-sided interval estimate. For example CLEVEL = 0.95 will give a 95% confidence interval.

Constraint: 0.0 < CLEVEL < 1.0.

4: PL – ***real***. *Output*

On exit: the lower limit, p_l, of the confidence interval.

5: PU – ***real***. *Output*

On exit: the upper limit, p_u, of the confidence interval.

6: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, $N < 1$,
or $K < 0$,
or $N < K$,
or CLEVEL ≤ 0.0,
or CLEVEL ≥ 1.0.

IFAIL = 2

When using the relationship with the gamma distribution to calculate one of the confidence limits, the series to calculate the gamma probabilities has failed to converge. Both PL and PU are set to zero. This is a very unlikely error exit and if it occurs please contact NAG.

7. Accuracy

For most cases using the beta deviates the results should have a relative accuracy of $\max(0.5E{-}12, 50.0\times\varepsilon)$ where ε is the ***machine precision*** (see X02AJF). Thus on machines with sufficiently high precision the results should be accurate to 12 significant figures. Some accuracy may be lost when $\alpha/2$ or $1-\alpha/2$ is very close to 0.0, which will occur if CLEVEL is very close to 1.0. This should not affect the usual confidence levels used.

The approximations used when n is large are accurate to at least 3 significant digits but usually to more.

8. Further Comments

None.

9. Example

The following example program reads in the number of deaths recorded among male recipients of war pensions in a six year period following an initial questionnaire in 1956. We consider two classes, non-smokers and those who reported that they smoked pipes only. The total number of males in each class is also read in. The data is taken from Snedecor and Cochran [2], page 216. An estimate of the probability of a death in the six year period in each class is computed together with 95% confidence intervals for these estimates.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07AAF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             CLEVEL, PHAT, PL, PU
      INTEGER          IFAIL, K, N
*     .. External Subroutines ..
      EXTERNAL         G07AAF
```

```
*     .. Intrinsic Functions ..
      INTRINSIC        real
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07AAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) ' Probability    Confidence Interval '
      WRITE (NOUT,*)
   20 READ (NIN,*,END=40) N, K, CLEVEL
      PHAT = real(K)/real(N)
      IFAIL = 0
*
      CALL G07AAF(N,K,CLEVEL,PL,PU,IFAIL)
*
      WRITE (NOUT,99999) PHAT, '( ', PL, ' , ', PU, ' )'
      GO TO 20
   40 STOP
*
99999 FORMAT (1X,F10.4,6X,A,F6.4,A,F6.4,A)
      END
```

9.2. Program Data

```
G07AAF Example Program Data
    1067      117    0.95           : N, K, CLEVEL
     402       54    0.95
```

9.3. Program Results

```
 G07AAF Example Program Results

  Probability    Confidence Interval

     0.1097      ( 0.0915 , 0.1300 )
     0.1343      ( 0.1025 , 0.1716 )
```

G07ABF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07ABF computes a confidence interval for the mean parameter of the Poisson distribution.

2. Specification

```
SUBROUTINE G07ABF (N, XMEAN, CLEVEL, TL, TU, IFAIL)
INTEGER        N, IFAIL
real           XMEAN, CLEVEL, TL, TU
```

3. Description

Given a random sample of size n, denoted by $x_1,x_2,\ldots,x_n$, from a Poisson distribution with probability function

$$p(x) = e^{-\theta}\frac{\theta^x}{x!}, \quad for x = 0,1,2,\ldots$$

the point estimate, $\hat{\theta}$, for θ is the sample mean, $\bar{x}$.

Given n and $\bar{x}$ this routine computes a $100(1-\alpha)\%$ confidence interval for the parameter θ, denoted by $[\theta_l,\theta_u]$, where α is in the interval $(0,1)$.

The lower and upper confidence limits are estimated by the solutions to the equations

$$e^{-n\theta_l}\sum_{x=T}^{\infty}\frac{(n\theta_l)^x}{x!} = \frac{\alpha}{2},$$

$$e^{-n\theta_u}\sum_{x=0}^{T}\frac{(n\theta_u)^x}{x!} = \frac{\alpha}{2},$$

where $T = \sum_{i=1}^{n} x_i = n\hat{\theta}.$

The relationship between the Poisson distribution and the χ^2 distribution (see Hastings and Peacock [1], page 112) is used to derive the equations

$$\theta_l = \frac{1}{2n}\chi^2_{2T,\alpha/2},$$

$$\theta_u = \frac{1}{2n}\chi^2_{2T+2,1-\alpha/2},$$

where $\chi^2_{\nu,p}$ is the deviate associated with the lower tail probability p of the χ^2 distribution with ν degrees of freedom.

In turn the relationship between the χ^2 distribution and the gamma distribution (see [1], page 70) yields the following equivalent equations;

$$\theta_l = \frac{1}{2n}\gamma_{T,2;\alpha/2},$$

$$\theta_u = \frac{1}{2n}\gamma_{T+1,2;1-\alpha/2},$$

where $\gamma_{\alpha,\beta;\delta}$ is the deviate associated with the lower tail probability, δ, of the gamma distribution with shape parameter α and scale parameter β. These deviates are computed using G01FFF.

4. References

[1] HASTINGS, N.A.J. and PEACOCK, J.B.
Statistical Distributions.
Butterworth, 1975.

[2] SNEDECOR, G.W. and COCHRAN, W.G.
Statistical Methods.
Iowa State University Press, 1967.

5. Parameters

1: N – INTEGER. ***Input***

On entry: the sample size, n.

Constraint: N $\geq$ 1.

2: XMEAN – ***real.*** ***Input***

On entry: the sample mean, $\bar{x}$.

Constraint: XMEAN $\geq$ 0.0.

3: CLEVEL – ***real.*** ***Input***

On entry: the confidence level, $(1-\alpha)$, for two-sided interval estimate. For example CLEVEL = 0.95 gives a 95% confidence interval.

Constraint: 0.0 < CLEVEL < 1.0.

4: TL – ***real.*** ***Output***

On exit: the lower limit, θ_l, of the confidence interval.

5: TU – ***real.*** ***Output***

On exit: the upper limit, θ_u, of the confidence interval.

6: IFAIL – INTEGER. ***Input/Output***

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, N < 1,
or XMEAN < 0.0,
or CLEVEL $\leq$ 0.0,
or CLEVEL $\geq$ 1.0.

IFAIL = 2

When using the relationship with the gamma distribution to calculate one of the confidence limits, the series to calculate the gamma probabilities has failed to converge. Both TL and TU are set to zero. This is a very unlikely error exit and if it occurs please contact NAG.

7. Accuracy

For most cases the results should have a relative accuracy of max(0.5E−12, 50.0×ε) where ε is the ***machine precision*** (see X02AJF). Thus on machines with sufficiently high precision the results should be accurate to 12 significant digits. Some accuracy may be lost when $\alpha/2$ or $1-\alpha/2$ is very close to 0.0, which will occur if CLEVEL is very close to 1.0. This should not affect the usual confidence intervals used.

8. Further Comments

None.

9. Example

The following example reads in data showing the number of noxious weed seeds and the frequency with which that number occured in 98 sub-samples of meadow grass. The data is taken from Snedecor and Cochran [2], page 224. The sample mean is computed as the point estimate of the Poisson parameter θ. The routine G07ABF is then called to compute both a 95% and a 99% confidence interval for the parameter θ.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07ABF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             CLEVEL, SUM, TL, TU, XMEAN
      INTEGER          I, IFAIL, IFREQ, N, NUM
*     .. External Subroutines ..
      EXTERNAL         G07ABF
*     .. Intrinsic Functions ..
      INTRINSIC        real
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07ABF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read in the number of Noxious Seeds in a sub sample and
*     the frequency with which that number occurs.
*
*     Compute the sample mean
*
      SUM = 0.0e0
      N = 0
   20 READ (NIN,*,END=40) NUM, IFREQ
      SUM = SUM + real(NUM)*real(IFREQ)
      N = N + IFREQ
      GO TO 20
   40 XMEAN = SUM/real(N)
      WRITE (NOUT,*)
      WRITE (NOUT,99999)
     +  'The point estimate of the Poisson parameter = ', XMEAN
*
      DO 60 I = 1, 2
         IF (I.EQ.1) THEN
            CLEVEL = 0.95e0
            WRITE (NOUT,*)
            WRITE (NOUT,*)
     +         '95 percent Confidence Interval for the estimate '
         ELSE
            CLEVEL = 0.99e0
            WRITE (NOUT,*)
            WRITE (NOUT,*)
     +         '99 percent Confidence Interval for the estimate '
         END IF
         IFAIL = 0
*
         CALL G07ABF(N,XMEAN,CLEVEL,TL,TU,IFAIL)
*
         WRITE (NOUT,99998) '( ', TL, ' , ', TU, ' )'
   60 CONTINUE
      STOP
*
99999 FORMAT (1X,A,F6.4)
99998 FORMAT (6X,A,F6.4,A,F6.4,A)
      END
```

9.2. Program Data

```
G07ABF Example Program Data
0 3
1 17
2 26
3 16
4 18
5 9
6 3
7 5
8 0
9 1
10 0
```

9.3. Program Results

```
 G07ABF Example Program Results

 The point estimate of the Poisson parameter = 3.0204

 95 percent Confidence Interval for the estimate
      ( 2.6861 , 3.3848 )

 99 percent Confidence Interval for the estimate
      ( 2.5874 , 3.5027 )
```

G07BBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07BBF computes maximum likelihood estimates and their standard errors for parameters of the Normal distribution from grouped and/or censored data.

2. Specification

```
SUBROUTINE G07BBF (METHOD, N, X, XC, IC, XMU, XSIG, TOL, MAXIT,
1                  SEXMU, SEXSIG, CORR, DEV, NOBS, NIT, WK,
2                  IFAIL)
INTEGER          N, IC(N), MAXIT, NOBS(4), NIT, IFAIL
real             X(N), XC(N), XMU, XSIG, TOL, SEXMU, SEXSIG, CORR,
1                DEV, WK(2*N)
CHARACTER*1      METHOD
```

3. Description

A sample of size n is taken from a Normal distribution with mean μ and variance σ^2 and consists of grouped and/or censored data. Each of the n observations is known by a pair of values (L_i,U_i) such that:

$$L_i \le x_i \le U_i$$

The data is represented as particular cases of this form:

exactly specified observations occur when $L_i = U_i = x_i$,
right-censored observations, known only by a lower bound, occur when $U_i \to \infty$,
left-censored observations, known only by a upper bound, occur when $L_i \to -\infty$,
and interval-censored observations when $L_i < x_i < U_i$.

Let the set A identify the exactly specified observations, sets B and C identify the observations censored on the right and left respectively, and set D identify the observations confined between two finite limits. Also let there be r exactly specified observations, i.e. the number in A. The probability density function for the standard Normal distribution is

$$Z(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right), \quad -\infty < x < \infty$$

and the cumulative distribution function is

$$P(X) = 1 - Q(X) = \int_{-\infty}^{X} Z(x)dx.$$

The log-likelihood of the sample can be written as:

$$L(\mu,\sigma) = -r\log\sigma - \frac{1}{2}\sum_A\{(x_i-\mu)/\sigma\}^2 + \sum_B\log(Q(l_i)) + \sum_C\log(P(u_i)) + \sum_D\log(p_i).$$

where $p_i = P(u_i) - P(l_i)$ and $u_i = (U_i-\mu)/\sigma, \quad l_i = (L_i-\mu)/\sigma$

Let

$$S(x_i) = \frac{Z(x_i)}{Q(x_i)}, \quad S_1(l_i,u_i) = \frac{Z(l_i) - Z(u_i)}{p_i}$$

and

$$S_2(l_i,u_i) = \frac{u_iZ(u_i) - l_iZ(l_i)}{p_i},$$

then the first derivatives of the log-likelihood can be written as:

$$\frac{\partial L(\mu,\sigma)}{\partial \mu} = L_1(\mu,\sigma) = \sigma^{-2}\sum_A (x_i-\mu) + \sigma^{-1}\sum_B S(l_i) - \sigma^{-1}\sum_C S(-u_i) + \sigma^{-1}\sum_D S_1(l_i,u_i)$$

and

$$\frac{\partial L(\mu,\sigma)}{\partial \sigma} = L_2(\mu,\sigma) = -r\sigma^{-1} + \sigma^{-3}\sum_A (x_i-\mu)^2 + \sigma^{-1}\sum_B l_i S(l_i) - \sigma^{-1}\sum_C u_i S(-u_i)$$
$$-\sigma^{-1}\sum_D S_2(l_i,u_i)$$

The maximum likelihood estimates, $\hat{\mu}$ and $\hat{\sigma}$, are the solution to the equations:

$$L_1(\hat{\mu},\hat{\sigma}) = 0 \qquad (1)$$

and

$$L_2(\hat{\mu},\hat{\sigma}) = 0 \qquad (2)$$

and if the second derivatives $\frac{\partial^2 L}{\partial \mu^2}$, $\frac{\partial^2 L}{\partial \mu \partial \sigma}$ and $\frac{\partial^2 L}{\partial \sigma^2}$ are denoted by L_{11}, L_{12} and L_{22} respectively, then estimates of the standard errors of $\hat{\mu}$ and $\hat{\sigma}$ are given by:

$$\mathrm{se}(\hat{\mu}) = \sqrt{\frac{-L_{22}}{L_{11}L_{22} - L_{12}^2}}, \qquad \mathrm{se}(\hat{\sigma}) = \sqrt{\frac{-L_{11}}{L_{11}L_{22} - L_{12}^2}}$$

and an estimate of the correlation of $\hat{\mu}$ and $\hat{\sigma}$ is given by:

$$\frac{L_{12}}{\sqrt{L_{12}L_{22}}}.$$

To obtain the maximum likelihood estimates the equations (1) and (2) can be solved using either the Newton-Raphson method or the Expectation-Maximization (*EM*) algorithm of Dempster *et al.* [1].

Newton-Raphson Method:

This consists of using approximate estimates $\tilde{\mu}$ and $\tilde{\sigma}$ to obtain improved estimates $\tilde{\mu} + \delta\tilde{\mu}$ and $\tilde{\sigma} + \delta\tilde{\sigma}$ by solving

$$\delta\tilde{\mu}L_{11} + \delta\tilde{\sigma}L_{12} + L_1 = 0,$$
$$\delta\tilde{\mu}L_{12} + \delta\tilde{\sigma}L_{22} + L_2 = 0,$$

for the corrections $\delta\tilde{\mu}$ and $\delta\tilde{\sigma}$.

EM Algorithm:

The expectation step consists of constructing the variable w_i as follows:

if $i \in A$, $\quad w_i = x_i$; (3)

if $i \in B$, $\quad w_i = E(x_i|x_i > L_i) = \mu + \sigma S(l_i)$; (4)

if $i \in C$, $\quad w_i = E(x_i|x_i < U_i) = \mu - \sigma S(-u_i)$; (5)

if $i \in D$, $\quad w_i = E(x_i|L_i < x_i < U_i) = \mu + \sigma S_1(l_i,u_i)$. (6)

the maximization step consists of substituting (3), (4), (5) and (6) into (1) and (2) giving:

$$\hat{\mu} = \sum_{i=1}^{n} \hat{w}_i / n \qquad (7)$$

and

$$\hat{\sigma}^2 = \sum_{i=1}^{n} (\hat{w}_i-\hat{\mu})^2 \Big/ \left\{ r + \sum_B T(\hat{l}_i) + \sum_C T(-\hat{u}_i) + \sum_D T_1(\hat{l}_i,\hat{u}_i) \right\} \qquad (8)$$

where

$$T(x) = S(x)\{S(x)-x\},\ T_1(l,u) = S_1^2(l,u) + S_2(l,u)$$

and where $\hat{w}_i$, $\hat{l}_i$ and $\hat{u}_i$ are w_i, l_i and u_i evaluated at $\hat{\mu}$ and $\hat{\sigma}$. Formulas (3) to (8) are the basis of the *EM* iterative procedure for finding $\hat{\mu}$ and $\hat{\sigma}^2$. The procedure consists of alternatively estimating $\hat{\mu}$ and $\hat{\sigma}^2$ using (7) and (8) and estimating $\{\hat{w}_i\}$ using (3) to (6).

In choosing between the two methods a general rule is that the Newton-Raphson method converges more quickly but requires good initial estimates whereas the *EM* algorithm converges slowly but is robust to the initial values. In the case of the censored Normal distribution, if only a small proportion of the observations are censored then estimates based on the exact observations should give good enough initial estimates for the Newton-Raphson method to be used. If there are a high proportion of censored observations then the *EM* algorithm should be used and if high accuracy is required the subsequent use of the Newton-Raphson method to refine the estimates obtained from the *EM* algorithm should be considered.

4. References

[1] DEMPSTER, A.P., LAIRD, N.M. and RUBIN, D.B.
Maximum Likelihood from Incomplete Data via the *EM* Algorithm (with Discussion).
J. R. Statist. Soc. B, 39, pp. 1-38, 1977.

[2] SWAN, A.V.
Algorithm AS16. Maximum Likelihood Estimation from Grouped and Censored Normal Data.
Appl. Statist., 18, pp. 110-114, 1969.

[3] WOLYNETZ, M.S.
Maximum Likelihood Estimation from Confined and Censored Normal Data.
Appl. Statist., 28, pp. 185-195, 1979.

5. Parameters

1: METHOD – CHARACTER*1. *Input*

On entry: indicates whether the Newton-Raphson or *EM* algorithm should be used.

If METHOD = 'N', then the Newton-Raphson algorithm is used.

If METHOD = 'E', then the *EM* algorithm is used.

Constraint: METHOD = 'N' or 'E'.

2: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N $\geq$ 2.

3: X(N) – ***real*** array. *Input*

On entry: the observations x_i, L_i or U_i, for $i = 1,2,...,n$.

If the observation is exactly specified – the exact value, x_i.

If the observation is right-censored – the lower value, L_i.

If the observation is left-censored – the upper value, U_i.

If the observation is interval-censored – the lower or upper value, L_i or U_i, (see XC).

4: XC(N) – ***real*** array. *Input*

On entry: if the jth observation, for $j = 1,2,...,n$ is an interval-censored observation then XC(j) should contain the complementary value to X(j), that is, if X(j) < XC(j), then XC(j) contains upper value, U_i, and if X(j) > XC(j), then XC(j) contains lower value, L_i.

Note if X(j) = XC(j) then the observation is ignored.

Otherwise if the jth observation is exact or right- or left-censored XC(j) need not be set.

5: IC(N) – INTEGER array. *Input*

On entry: IC(i) contains the censoring codes for the ith observation, for $i = 1,2,...,n$.

If IC(i) $= 0$ the observation is exactly specified.

If IC(i) $= 1$ the observation is right-censored.

If IC(i) $= 2$ the observation is left-censored.

If IC(i) $= 3$ the observation is interval-censored.

Constraint: IC(i) $= 0$, 1, 2 or 3, for $i = 1,2,...,n$.

6: XMU – ***real.*** *Input/Output*

On entry: if XSIG > 0.0 the initial estimate of the mean, μ; otherwise XMU need not be set.

On exit: the maximum likelihood estimate, $\hat{\mu}$, of μ.

7: XSIG – ***real.*** *Input/Output*

On entry: specifies whether an initial estimate of μ and σ are to be supplied. If XSIG > 0.0, then XSIG is the initial estimate of σ and XMU must contain an initial estimate of μ.

If XSIG ≤ 0.0, then initial estimates of XMU and XSIG are calculated internally from:

(a) the exact observations, if the number of exactly specified observations is ≥ 2; or
(b) the interval-censored observations; if the number of interval-censored observations is ≥ 1; or
(c) they are set to 0.0 and 1.0 respectively.

On exit: the maximum likelihood estimate, $\hat{\sigma}$, of σ.

8: TOL – ***real.*** *Input*

On entry: the relative precision required for the final estimates of μ and σ. Convergence is assumed when the absolute relative changes in the estimates of both μ and σ are less than TOL.

If TOL $= 0.0$, then a relative precision of 0.000005 is used.

Constraint: ***machine precision*** $<$ TOL ≤ 1.0 or TOL $= 0.0$.

9: MAXIT – INTEGER. *Input*

On entry: the maximum number of iterations.

If MAXIT ≤ 0, then a value of 25 is used.

10: SEXMU – ***real.*** *Output*

On exit: the estimate of the standard error of $\hat{\mu}$.

11: SEXSIG – ***real.*** *Output*

On exit: the estimate of the standard error of $\hat{\sigma}$.

12: CORR – ***real.*** *Output*

On exit: the estimate of the correlation between $\hat{\mu}$ and $\hat{\sigma}$.

13: DEV – ***real.*** *Output*

On exit: the maximized log-likelihood, $L(\hat{\mu},\hat{\sigma})$.

14: NOBS(4) – INTEGER array. *Output*

On exit: the number of the different types of each observation;

NOBS(1) contains number of right-censored observations.
NOBS(2) contains number of left-censored observations.

NOBS(3) contains number of interval-censored observations.
NOBS(4) contains number of exactly specified observations.

15: NIT – INTEGER. *Output*

On exit: the number of iterations performed.

16: WK(2*N) – ***real*** array. *Workspace*

17: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, METHOD ≠ 'N' or 'E',
or N < 2,
or IC(i) ≠ 0, 1, 2 or 3, for some i,
or TOL < 0.0,
or 0.0 < TOL < ***machine precision***,
or TOL > 1.0.

IFAIL = 2

The chosen method failed to converge in MAXIT iterations. The user should either increase TOL or MAXIT or, if using the *EM* algorithm try using the Newton-Raphson method with initial values those returned by the current call to G07BBF. All returned values will be reasonable approximations to the correct results if MAXIT is not very small.

IFAIL = 3

The chosen method is diverging. This will be due to poor initial values. The user should try different initial values.

IFAIL = 4

G07BBF was unable to calculate the standard errors. This can be caused by the method starting to diverge when the maximum number of iterations was reached.

7. Accuracy

The accuracy is controlled by the parameter TOL.

If high precision is requested with the *EM* algorithm then there is a possibility that, due to the slow convergence, before the correct solution has been reached the increments of $\hat{\mu}$ and $\hat{\sigma}$ may be smaller than TOL and the process will prematurely assume convergence.

8. Further Comments

The process is deemed divergent if three successive increments of μ or σ increase.

9. Example

A sample of 18 observations and their censoring codes are read in and the Newton-Raphson method used to compute the estimates.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07BBF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=20)
*     .. Local Scalars ..
      real             CORR, DEV, SEXMU, SEXSIG, TOL, XMU, XSIG
      INTEGER          I, IFAIL, MAXIT, N, NIT
      CHARACTER        METHOD
*     .. Local Arrays ..
      real             WK(2*NMAX), X(NMAX), XC(NMAX)
      INTEGER          IC(NMAX), NOBS(4)
*     .. External Subroutines ..
      EXTERNAL         G07BBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07BBF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, METHOD, XMU, XSIG, TOL, MAXIT
      IF (N.LE.NMAX) THEN
         READ (NIN,*) (X(I),XC(I),IC(I),I=1,N)
         IFAIL = 0
*
         CALL G07BBF(METHOD,N,X,XC,IC,XMU,XSIG,TOL,MAXIT,SEXMU,SEXSIG,
     +               CORR,DEV,NOBS,NIT,WK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) ' Mean = ', XMU
         WRITE (NOUT,99999) ' Standard deviation = ', XSIG
         WRITE (NOUT,99999) ' Standard error of mean = ', SEXMU
         WRITE (NOUT,99999) ' Standard error of sigma = ', SEXSIG
         WRITE (NOUT,99999) ' Correlation coefficient = ', CORR
         WRITE (NOUT,99998) ' Number of right censored observations = ',
     +     NOBS(1)
         WRITE (NOUT,99998) ' Number of left censored observations = ',
     +     NOBS(2)
         WRITE (NOUT,99998)
     +     ' Number of interval censored observations = ', NOBS(3)
         WRITE (NOUT,99998)
     +     ' Number of exactly specified observations = ', NOBS(4)
         WRITE (NOUT,99998) ' Number of iterations = ', NIT
         WRITE (NOUT,99999) ' Log-likelihood = ', DEV
      END IF
      STOP
*
99999 FORMAT (1X,A,F8.4)
99998 FORMAT (1X,A,I2)
      END
```

9.2. Program Data

```
G07BBF Example Program Data
18 'N' 4.0 1.0 0.00005 50
4.5 0.0 0 5.4 0.0 0 3.9 0.0 0 5.1 0.0 0 4.6 0.0 0 4.8 0.0 0
2.9 0.0 0 6.3 0.0 0 5.5 0.0 0 4.6 0.0 0 4.1 0.0 0 5.2 0.0 0
3.2 0.0 1 4.0 0.0 1 3.1 0.0 1 5.1 0.0 2 3.8 0.0 2 2.2 2.5 3
```

9.3. Program Results

```
G07BBF Example Program Results

 Mean =    4.4924
 Standard deviation =   1.0196
 Standard error of mean =   0.2606
 Standard error of sigma =   0.1940
 Correlation coefficient =   0.0160
 Number of right censored observations =  3
 Number of left censored observations =  2
 Number of interval censored observations =  1
 Number of exactly specified observations = 12
 Number of iterations =  5
 Log-likelihood = -22.2817
```

G07BEF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07BEF computes maximum likelihood estimates for parameters of the Weibull distribution from data which may be right-censored.

2. Specification

```
      SUBROUTINE G07BEF (CENS, N, X, IC, BETA, GAMMA, TOL, MAXIT, SEBETA,
     1                   SEGAM, CORR, DEV, NIT, WK, IFAIL)
      INTEGER            N, IC(*), MAXIT, NIT, IFAIL
      real               X(N), BETA, GAMMA, TOL, SEBETA, SEGAM, CORR, DEV, WK(N)
      CHARACTER*1        CENS
```

3. Description

G07BEF computes maximum likelihood estimates of the parameters of the Weibull distribution from exact or right-censored data.

For n realizations, y_i, from a Weibull distribution a value x_i is observed such that

$$x_i \leq y_i.$$

There are two situations:

(a) exactly specified observations, when $x_i = y_i$

(b) right-censored observations, known by a lower bound, when $x_i < y_i$.

The probability density function of the Weibull distribution, and hence the contribution of an exactly specified observation to the likelihood, is given by:

$$f(x;\lambda,\gamma) = \lambda\gamma x^{\gamma-1}\exp(-\lambda x^{\gamma}), \quad x > 0, \quad \text{for } \lambda,\gamma > 0;$$

while the survival function of the Weibull distribution, and hence the contribution of a right-censored observation to the likelihood, is given by:

$$S(x;\lambda,\gamma) = \exp(-\lambda x^{\gamma}), \quad x > 0, \quad \text{for } \lambda,\gamma > 0.$$

If d of the n observations are exactly specified and indicated by $i \in D$ and the remaining $(n-d)$ are right-censored, then the likelihood function, $Like(\lambda,\gamma)$ is given by

$$Like(\lambda,\gamma) \propto (\lambda\gamma)^d\left(\prod_{i\in D} x_i^{\gamma-1}\right)\exp\left(-\lambda\sum_{i=1}^{n} x_i^{\gamma}\right).$$

To avoid possible numerical instability a different parameterization β,γ is used, with $\beta = \log(\lambda)$. The kernel log-likelihood function, $L(\beta,\gamma)$, is then:

$$L(\beta,\gamma) = d\log(\gamma) + d\beta + (\gamma-1)\sum_{i\in D}\log(x_i) - e^{\beta}\sum_{i=1}^{n} x_i^{\gamma}.$$

If the derivatives $\frac{\partial L}{\partial\beta}$, $\frac{\partial L}{\partial\gamma}$, $\frac{\partial^2 L}{\partial\beta^2}$, $\frac{\partial^2 L}{\partial\beta\partial\gamma}$ and $\frac{\partial^2 L}{\partial\gamma^2}$ are denoted by L_1, L_2, L_{11}, L_{12} and L_{22}, respectively, then the maximum likelihood estimates, $\hat{\beta}$ and $\hat{\gamma}$, are the solution to the equations:

$$L_1(\hat{\beta},\hat{\gamma}) = 0 \qquad (1)$$

and

$$L_2(\hat{\beta},\hat{\gamma}) = 0. \qquad (2)$$

Estimates of the asymptotic standard errors of $\hat{\beta}$ and $\hat{\gamma}$ are given by:

$$\text{se}(\hat{\beta}) = \sqrt{\frac{-L_{22}}{L_{11}L_{22}-L_{12}^2}}, \qquad \text{se}(\hat{\gamma}) = \sqrt{\frac{-L_{11}}{L_{11}L_{22}-L_{12}^2}}.$$

An estimate of the correlation coefficient of $\hat{\beta}$ and $\hat{\gamma}$ is given by:

$$\frac{L_{12}}{\sqrt{L_{12}L_{22}}}.$$

Note: if an estimate of the original parameter λ is required, then

$$\hat{\lambda} = \exp(\hat{\beta}) \quad \text{and} \quad \mathrm{se}(\hat{\lambda}) = \hat{\lambda}\mathrm{se}(\hat{\beta}).$$

The equations (1) and (2) are solved by the Newton-Raphson iterative method with adjustments made to ensure that $\hat{\gamma} > 0.0$.

4. References

[1] GROSS, A.J. and CLARK, V.A.
Survival Distributions: Reliability Applications in the Biomedical Sciences.
Wiley, 1975.

[2] KALBFLEISCH, J.D. and PRENTICE, R.L.
The Statistical Analysis of Failure Time Data.
Wiley, 1980.

5. Parameters

1: CENS – CHARACTER*1. *Input*

On entry: indicates whether the data is censored or non-censored.

If CENS = 'N', then each observation is assumed to be exactly specified. IC is not referenced.

If CENS = 'C', then each observation is censored according to the value contained in IC(i), for $i = 1,2,...,n$.

Constraint: CENS = 'C' or 'N'.

2: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N $\geq$ 1.

3: X(N) – ***real*** array. *Input*

On entry: X(i) contains the ith observation, x_i, for $i = 1,2,...,n$.

Constraint: X(i) > 0.0, for $i = 1,2,...,n$.

4: IC(*) – INTEGER array. *Input*

Note: if CENS = 'C', then IC must be dimensioned at least N, otherwise IC can be dimensioned 1.

On entry: if CENS = 'C', then IC(i) contains the censoring codes for the ith observation, for $i = 1,2,...,n$.

If IC(i) = 0, the ith observation is exactly specified.

If IC(i) = 1, the ith observation is right-censored.

If CENS = 'N', then IC is not referenced.

Constraint: if CEN = 'C', then IC(i) = 0 or 1, for $i = 1,2,...,n$.

5: BETA – ***real***. *Output*

On exit: the maximum likelihood estimate, $\hat{\beta}$, of β.

6: GAMMA – ***real.*** *Input/Output*

On entry: indicates whether an initial estimate of γ is provided.

If GAMMA > 0.0, it is taken as the initial estimate of γ and an initial estimate of β is calculated from this value of γ.

If GAMMA ≤ 0.0, then initial estimates of γ and β are calculated, internally, providing the data contains at least two distinct exact observations. (If there are only two distinct exact observations, then the largest observation must not be exactly specified.) See Section 8 for further details.

On exit: contains the maximum likelihood estimate, $\hat{\gamma}$, of γ.

7: TOL – ***real.*** *Input*

On entry: the relative precision required for the final estimates of β and γ. Convergence is assumed when the absolute relative changes in the estimates of both β and γ are less than TOL.

If TOL = 0.0, then a relative precision of 0.000005 is used.

Constraint: ***machine precision*** ≤ TOL ≤ 1.0 or TOL = 0.0.

8: MAXIT – INTEGER. *Input*

On entry: the maximum number of iterations allowed.

If MAXIT ≤ 0, then a value of 25 is used.

9: SEBETA – ***real.*** *Output*

On exit: an estimate of the standard error of $\hat{\beta}$.

10: SEGAM – ***real.*** *Output*

On exit: an estimate of the standard error of $\hat{\gamma}$.

11: CORR – ***real.*** *Output*

On exit: an estimate of the correlation between $\hat{\beta}$ and $\hat{\gamma}$.

12: DEV – ***real.*** *Output*

On exit: the maximized kernel log-likelihood, $L(\hat{\beta},\hat{\gamma})$.

13: NIT – INTEGER. *Output*

On exit: the number of iterations performed.

14: WK(N) – ***real*** array. *Workspace*

15: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, CENS ≠ 'N' or 'C',
or N < 1,
or TOL < 0.0,
or 0.0 < TOL < ***machine precision***,
or TOL > 1.0.

IFAIL = 2

On entry, the ith observation, $X(i) \leq 0.0$, for some $i = 1,2,...,n$,
or the ith censoring code, $IC(i) \neq 0$ or 1, for some $i = 1,2,...,n$ and CENS = 'C'.

IFAIL = 3

On entry, there are no exactly specified observations, or the routine was requested to calculate initial values and there are either less than two distinct exactly specified observations or there are exactly two and the largest observation is one of the exact observations.

IFAIL = 4

The method has failed to converge in MAXIT iterations. The user should increase TOL or MAXIT.

IFAIL = 5

Process has diverged. The process is deemed divergent if three successive increments of β or γ increase or if the Hessian matrix of the Newton-Raphson process is singular. Either different initial estimates should be provided or the data should be checked to see if the Weibull distribution is appropriate.

IFAIL = 6

A potential overflow has been detected. This is an unlikely exit usually caused by a large input estimate of γ.

7. Accuracy

Given that the Weibull distribution is a suitable model for the data and that the initial values are reasonable the convergence to the required accuracy, indicated by TOL, should be achieved.

8. Further Comments

The initial estimate of γ is found by calculating a Kaplan-Meier estimate of the survival function, $\hat{S}(x)$, and estimating the gradient of the plot of $\log(-\log(\hat{S}(x)))$ against x. This requires the Kaplan-Meier estimate to have at least two distinct points.

The initial estimate of $\hat{\beta}$, given a value of $\hat{\gamma}$, is calculated as

$$\hat{\beta} = \log\left(\frac{d}{\sum_{i=1}^{n} x_i^{\hat{\gamma}}}\right).$$

9. Example

In a study, 20 patients receiving an analgesic to relieve headache pain had the following recorded relief times (in hours): 1.1 1.4 1.3 1.7 1.9 1.8 1.6 2.2 1.7 2.7 4.1 1.8 1.5 1.2 1.4 3.0 1.7 2.3 1.6 2.0 (See Gross and Clarke [1]). This data is read in and a Weibull distribution fitted assuming no censoring; the parameter estimates and their standard errors are printed.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07BEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=20)
*     .. Local Scalars ..
      real             BETA, CORR, DEV, GAMMA, SEBETA, SEGAM, TOL
      INTEGER          I, IFAIL, MAXIT, N, NIT
*     .. Local Arrays ..
      real             WK(NMAX), X(NMAX)
      INTEGER          IC(NMAX)
*     .. External Subroutines ..
      EXTERNAL         G07BEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07BEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) (X(I),I=1,N)
*        If data were censored then IC would also be read in.
*        Leave G07BEF to calculate initial values
         GAMMA = 0.0e0
*        Use default values for TOL and MAXIT
         TOL = 0.0e0
         MAXIT = 0
         IFAIL = 0
*
         CALL G07BEF('No censor',N,X,IC,BETA,GAMMA,TOL,MAXIT,SEBETA,
     +               SEGAM,CORR,DEV,NIT,WK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) ' BETA  = ', BETA, ' Standard error = ',
     +     SEBETA
         WRITE (NOUT,99999) ' GAMMA = ', GAMMA, ' Standard error = ',
     +     SEGAM
      END IF
      STOP
*
99999 FORMAT (1X,2(A,F10.4))
      END
```

9.2. Program Data

```
G07BEF Example Program Data
20
1.1 1.4 1.3 1.7 1.9 1.8 1.6 2.2 1.7 2.7
4.1 1.8 1.5 1.2 1.4 3.0 1.7 2.3 1.6 2.0
```

9.3. Program Results

```
 G07BEF Example Program Results

  BETA  =    -2.1073 Standard error =     0.4627
  GAMMA =     2.7870 Standard error =     0.4273
```

G07CAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07CAF computes a t-test statistic to test for a difference in means between two Normal populations, together with a confidence interval for the difference between the means.

2. Specification

```
SUBROUTINE G07CAF (TAIL, EQUAL, NX, NY, XMEAN, YMEAN, XSTD, YSTD,
1                  CLEVEL, T, DF, PROB, DL, DU, IFAIL)
INTEGER        NX, NY, IFAIL
real           XMEAN, YMEAN, XSTD, YSTD, CLEVEL, T, DF, PROB, DL, DU
CHARACTER*1    TAIL, EQUAL
```

3. Description

Consider two independent samples, denoted by X and Y, of size n_x and n_y drawn from two Normal populations with means μ_x and μ_y, and variances σ_x^2 and σ_y^2 respectively. Denote the sample means by $\bar{x}$ and $\bar{y}$ and the sample variances by s_x^2 and s_y^2 respectively.

G07CAF calculates a test statistic and its significance level to test the null hypothesis $H_0 : \mu_x = \mu_y$, together with upper and lower confidence limits for $\mu_x - \mu_y$. The test used depends on whether or not the two population variances are assumed to be equal.

(1) It is assumed that the two variances are equal, that is $\sigma_x^2 = \sigma_y^2$.

The test used is the two sample t-test. The test statistic t is defined by;

$$t_{obs} = \frac{\bar{x} - \bar{y}}{s\sqrt{(1/n_x)+(1/n_y)}}$$

where $s^2 = \dfrac{(n_x-1)s_x^2 + (n_y-1)s_y^2}{n_x + n_y - 2}$ is the pooled variance of the two samples.

Under the null hypothesis H_0 this test statistic has a t-distribution with (n_x+n_y-2) degrees of freedom.

The test of H_0 is carried out against one of three possible alternatives;

$H_1 : \mu_x \neq \mu_y$; the significance level, $p = P(t \geq |t_{obs}|)$, i.e. a two-tailed probability.

$H_1 : \mu_x > \mu_y$; the significance level, $p = P(t \geq t_{obs})$, i.e. an upper tail probability.

$H_1 : \mu_x < \mu_y$; the significance level, $p = P(t \leq t_{obs})$, i.e. a lower tail probability.

Upper and lower $100(1-\alpha)\%$ confidence limits for $\mu_x - \mu_y$ are calculated as:

$$(\bar{x}-\bar{y}) \pm t_{1-\alpha/2}s\sqrt{(1/n_x)+(1/n_y)},$$

where $t_{1-\alpha/2}$ is the $100(1-\alpha/2)$ percentage point of the t-distribution with (n_x+n_y-2) degrees of freedom.

(2) It is not assumed that the two variances are equal.

If the population variances are not equal the usual two sample t-statistic no longer has a t-distribution and an approximate test is used.

This problem is often referred to as the Behrens-Fisher problem, see Kendall and Stuart [2]. The test used here is based on Satterthwaites procedure. To test the null hypothesis the test statistic t' is used where

$$t'_{obs} = \frac{\bar{x}-\bar{y}}{\mathrm{se}(\bar{x}-\bar{y})}$$

where $\text{se}(\bar{x}-\bar{y}) = \sqrt{\frac{s_x^2}{n_x} + \frac{s_y^2}{n_y}}$.

A t-distribution with f degrees of freedom is used to approximate the distribution of t'

$$\text{where } f = \frac{\text{se}(\bar{x}-\bar{y})^4}{\frac{s_x^2/n_x^2}{(n_x-1)} + \frac{s_y^2/n_y^2}{(n_y-1)}}.$$

The test of H_0 is carried out against one of the three alternative hypotheses described above, replacing t by t' and t_{obs} by t'_{obs}.

Upper and lower $100(1-\alpha)\%$ confidence limits for $\mu_x-\mu_y$ are calculated as:

$$(\bar{x}-\bar{y}) \pm t_{1-\alpha/2}\text{se}(\bar{x}-\bar{y}),$$

where $t_{1-\alpha/2}$ is the $100(1-\alpha/2)$ percentage point of the t-distribution with f degrees of freedom.

4. References

[1] JOHNSON, M.G. and KOTZ, A.
The Encyclopedia of Statistics, Volume 2.
Griffin, 1969.

[2] KENDALL, M.G. and STUART, A.
The Advanced Theory of Statistics, Volume 2.
Griffin, 1969.

[3] SNEDECOR, G.W. and COCHRAN, W.G.
Statistical Methods.
Iowa State University Press, 1967.

5. Parameters

1: TAIL – CHARACTER*1. *Input*

On entry: indicates which tail probability is to be calculated, and thus which alternative hypothesis is to be used.

If TAIL = 'T', the two tail probability, i.e. $H_1 : \mu_x \neq \mu_y$.

If TAIL = 'U', the upper tail probability, i.e. $H_1 : \mu_x > \mu_y$.

If TAIL = 'L', the lower tail probability, i.e. $H_1 : \mu_x < \mu_y$.

Constraint: TAIL = 'T', 'U' or 'L'.

2: EQUAL – CHARACTER*1. *Input*

On entry: indicates whether the population variances are assumed to be equal or not.

If EQUAL = 'E', the population variances are assumed to be equal, that is $\sigma_x^2 = \sigma_y^2$.

If EQUAL = 'U', the population variances are not assumed to be equal.

Constraint: EQUAL = 'E' or 'U'.

3: NX – INTEGER. *Input*

On entry: the size of the X sample, n_x.

Constraint: NX $\geq$ 2.

4: NY – INTEGER. *Input*

On entry: the size of the Y sample, n_y.

Constraint: NY $\geq$ 2.

5: XMEAN – ***real.*** *Input*

On entry: the mean of the X sample, $\bar{x}$.

6: YMEAN – ***real.*** *Input*

On entry: the mean of the Y sample, $\bar{y}$.

7: XSTD – ***real.*** *Input*

On entry: the standard deviation of the X sample, s_x.

Constraint: XSTD > 0.0.

8: YSTD – ***real.*** *Input*

On entry: the standard deviation of the Y sample, s_y.

Constraint: YSTD > 0.0.

9: CLEVEL – ***real.*** *Input*

On entry: the confidence level, $1 - \alpha$, for the specified tail. For example CLEVEL = 0.95 will give a 95% confidence interval.

Constraint: 0.0 < CLEVEL < 1.0.

10: T – ***real.*** *Output*

On exit: contains the test statistic, t_{obs} or t'_{obs}.

11: DF – ***real.*** *Output*

On exit: contains the degrees of freedom for the test statistic.

12: PROB – ***real.*** *Output*

On exit: contains the significance level, that is the tail probability, p, as defined by TAIL.

13: DL – ***real.*** *Output*

On exit: contains the lower confidence limit for $\mu_x - \mu_y$.

14: DU – ***real.*** *Output*

On exit: contains the upper confidence limit for $\mu_x - \mu_y$.

15: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, TAIL ≠ 'T', 'U' or 'L',
or EQUAL ≠ 'E' or 'U',
or NX < 2,
or NY < 2,
or XSTD ≤ 0.0,

or YSTD ≤ 0.0,
or CLEVEL ≤ 0.0,
or CLEVEL ≥ 1.0.

7. Accuracy

The computed probability and the confidence limits should be accurate to approximately 5 significant figures.

8. Further Comments

The sample means and standard deviations can be computed using G01AAF.

9. Example

The following example program reads the two sample sizes and the sample means and standard deviations for two independent samples. The data is taken from Snedecor and Cochran, page 116, from a test to compare two methods of estimating the concentration of a chemical in a vat. A test of the equality of the means is carried out first assuming that the two population variances are equal and then making no assumption about the equality of the population variances.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07CAF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             CLEVEL, DF, DL, DU, PROB, T, XMEAN, XSTD, YMEAN,
     +                 YSTD
      INTEGER          IFAIL, NX, NY
*     .. External Subroutines ..
      EXTERNAL         G07CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07CAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) NX, NY
      READ (NIN,*) XMEAN, YMEAN, XSTD, YSTD
      READ (NIN,*) CLEVEL
      IFAIL = 0
*
      CALL G07CAF('Two','Equal',NX,NY,XMEAN,YMEAN,XSTD,YSTD,CLEVEL,T,DF,
     +            PROB,DL,DU,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Assuming population variances are equal.'
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 't test statistic = ', T
      WRITE (NOUT,99998) 'Degrees of freedom = ', DF
      WRITE (NOUT,99997) 'Significance level = ', PROB
      WRITE (NOUT,99999)
     + 'Lower confidence limit for difference in means = ', DL
      WRITE (NOUT,99999)
     + 'Upper confidence limit for difference in means = ', DU
      WRITE (NOUT,*)
      IFAIL = 0
*
```

```
      CALL G07CAF('Two','Unequal',NX,NY,XMEAN,YMEAN,XSTD,YSTD,CLEVEL,T,
     +            DF,PROB,DL,DU,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'No assumptions about population variances .'
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 't test statistic = ', T
      WRITE (NOUT,99997) 'Degrees of freedom = ', DF
      WRITE (NOUT,99997) 'Significance level = ', PROB
      WRITE (NOUT,99999)
     + 'Lower confidence limit for difference in means = ', DL
      WRITE (NOUT,99999)
     + 'Upper confidence limit for difference in means = ', DU
      STOP
*
99999 FORMAT (1X,A,F10.4)
99998 FORMAT (1X,A,F8.1)
99997 FORMAT (1X,A,F8.4)
      END
```

9.2. Program Data

```
G07CAF Example Program Data
 4  8
 25.0 21.0
 0.8185  4.2083
 0.95
```

9.3. Program Results

```
 G07CAF Example Program Results

 Assuming population variances are equal.

 t test statistic =     1.8403
 Degrees of freedom =     10.0
 Significance level =   0.0955
 Lower confidence limit for difference in means =    -0.8429
 Upper confidence limit for difference in means =     8.8429


 No assumptions about population variances .

 t test statistic =     2.5922
 Degrees of freedom =   7.9925
 Significance level =   0.0320
 Lower confidence limit for difference in means =     0.4410
 Upper confidence limit for difference in means =     7.5590
```

G07DAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07DAF finds the median, median absolute deviation, and a robust estimate of the standard deviation for a set of ungrouped data.

2. Specification

```
SUBROUTINE G07DAF (N, X, Y, XME, XMD, XSD, IFAIL)
INTEGER        N, IFAIL
real           X(N), Y(N), XME, XMD, XSD
```

3. Description

The data consists of a sample of size n, denoted by $x_1,x_2,...,x_n$, drawn from a random variable X.

G07DAF first computes the median,

$$\theta_{med} = \underset{i}{\mathrm{med}}\{x_i\},$$

and from this the median absolute deviation can be computed,

$$\sigma_{med} = \underset{i}{\mathrm{med}}\{|x_i-\theta_{med}|\}.$$

Finally, a robust estimate of the standard deviation is computed,

$$\sigma'_{med} = \sigma_{med}/\Phi^{-1}(0.75)$$

where $\Phi^{-1}(0.75)$ is the value of the inverse standard Normal function at the point 0.75.

G07DAF is based upon subroutine LTMDDV within the ROBETH library, see Marazzi [2].

4. References

[1] HUBER, P.J.
Robust Statistics.
Wiley, 1981.

[2] MARAZZI, A.
Subroutines for Robust Estimation of Location and Scale in ROBETH.
Institut Universitaire de Médecine Sociale et Préventive, Lausanne, 1987.
(Cah Rech Doc IUMSP, No 3 ROB 1).

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N > 1.

2: X(N) – ***real*** array. *Input*

On entry: the vector of observations, $x_1,x_2,...,x_n$.

3: Y(N) – ***real*** array. *Output*

On exit: the observations sorted into ascending order.

4: XME – ***real***. *Output*

On exit: the median, θ_{med}.

5: XMD – ***real.*** *Output*

On exit: the median absolute deviation, σ_{med}.

6: XSD – ***real.*** *Output*

On exit: the robust estimate of the standard deviation, σ'_{med}.

7: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, N $\le$ 1.

7. Accuracy

The computations are believed to be stable.

8. Further Comments

Unless otherwise stated in the implementation document, the routine may be called with the same actual array supplied for parameters X and Y, in which case the sorted data values will overwrite the original contents of X. However this is not standard Fortran 77, and may not work on all systems.

9. Example

The following program reads in a set of data consisting of eleven observations of a variable *X*. The median, median absolute deviation and a robust estimate of the standard deviation are calculated and printed along with the sorted data in output array Y.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07DAF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=25)
*     .. Local Scalars ..
      real             XMD, XME, XSD
      INTEGER          I, IFAIL, N
*     .. Local Arrays ..
      real             X(NMAX), Y(NMAX)
*     .. External Subroutines ..
      EXTERNAL         G07DAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07DAF Example Program Results'
```

```
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      WRITE (NOUT,*)
      IF (N.LE.NMAX) THEN
         READ (NIN,*) (X(I),I=1,N)
         IFAIL = 0
*
         CALL G07DAF(N,X,Y,XME,XMD,XSD,IFAIL)
*
         WRITE (NOUT,*) 'Output Y:'
         WRITE (NOUT,99999) (Y(I),I=1,N)
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'XME = ', XME, ', XMD = ', XMD, ', XSD = ',
     +     XSD
      ELSE
         WRITE (NOUT,99997) 'N is out of range: N =', N
      END IF
      STOP
*
99999 FORMAT (1X,11F7.3)
99998 FORMAT (1X,A,F6.3,A,F6.3,A,F6.3)
99997 FORMAT (1X,A,I5)
      END
```

9.2. Program Data

```
G07DAF Example Program Data
11          : N, NUMBER OF OBSERVATIONS
13.0 11.0 16.0 5.0 3.0 18.0 9.0 8.0 6.0 27.0 7.0  : X, OBSERVATIONS
```

9.3. Program Results

```
 G07DAF Example Program Results

 Output Y:
   3.000  5.000  6.000  7.000  8.000  9.000 11.000 13.000
16.000 18.000 27.000

 XME =  9.000, XMD =  4.000, XSD =  5.930
```

G07DBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07DBF computes an M-estimate of location with (optional) simultaneous estimation of the scale using Huber's algorithm.

2. Specification

```
      SUBROUTINE G07DBF (ISIGMA, N, X, IPSI, C, H1, H2, H3, DCHI, THETA,
     1                   SIGMA, MAXIT, TOL, RS, NIT, WRK, IFAIL)
      INTEGER          ISIGMA, N, IPSI, MAXIT, NIT, IFAIL
      real             X(N), C, H1, H2, H3, DCHI, THETA, SIGMA, TOL,
     1                 RS(N), WRK(N)
```

3. Description

The data consists of a sample of size n, denoted by $x_1,x_2,\ldots,x_n$, drawn from a random variable X.

The x_i are assumed to be independent with an unknown distribution function of the form

$$F((x_i-\theta)/\sigma)$$

where θ is a location parameter, and σ is a scale parameter. M-estimators of θ and σ are given by the solution to the following system of equations:

$$\sum_{i=1}^{n} \psi\left(\left(x_i-\hat{\theta}\right)/\hat{\sigma}\right) = 0 \qquad (1)$$

$$\sum_{i=1}^{n} \chi\left(\left(x_i-\hat{\theta}\right)/\hat{\sigma}\right) = (n-1)\beta \qquad (2)$$

where ψ and χ are given functions, and β is a constant, such that $\hat{\sigma}$ is an unbiased estimator when x_i, for $i = 1,2,\ldots,n$ has a normal distribution. Optionally, the second equation can be omitted and the first equation is solved for $\hat{\theta}$ using an assigned value of $\sigma = \sigma_c$.

The values of $\psi\left(\frac{x_i-\hat{\theta}}{\hat{\sigma}}\right)\hat{\sigma}$ are known as the Winsorized residuals.

The following functions are available for ψ and χ in G07DBF;

(a) **Null Weights**

$$\psi(t) = t \qquad\qquad \chi(t) = \frac{t^2}{2}$$

Use of these null functions leads to the mean and standard deviation of the data.

(b) **Huber's Function**

$$\psi(t) = \max(-c,\min(c,t)) \qquad \chi(t) = \frac{|t|^2}{2} \quad |t| \le d$$

$$\chi(t) = \frac{d^2}{2} \quad |t| > d$$

(c) **Hampel's Piecewise Linear Function**

$$\psi_{h_1,h_2,h_3}(t) = -\psi_{h_1,h_2,h_3}(-t)$$

$$= t \qquad 0 \le t \le h_1 \qquad\qquad \chi(t) = \frac{|t|^2}{2} \quad |t| \le d$$

$$\begin{aligned} &= h_1 && h_1 \le t \le h_2 \\ &= h_1(h_3-t)/(h_3-h_2) && h_2 \le t \le h_3 \\ &= 0 && t > h_3 \end{aligned} \qquad \chi(t) = \frac{d^2}{2} \quad |t| > d$$

(d) **Andrew's Sine Wave Function**

$$\begin{aligned} \psi(t) &= \sin t && -\pi \le t \le \pi \qquad & \chi(t) &= \frac{|t|^2}{2} \quad |t| \le d \\ &= 0 && \text{otherwise} & \chi(t) &= \frac{d^2}{2} \quad |t| > d \end{aligned}$$

(e) **Tukey's Bi-weight**

$$\begin{aligned} \psi(t) &= t(1-t^2)^2 && |t| \le 1 \qquad & \chi(t) &= \frac{|t|^2}{2} \quad |t| \le d \\ &= 0 && \text{otherwise} & \chi(t) &= \frac{d^2}{2} \quad |t| > d \end{aligned}$$

where c, h_1, h_2, h_3 and d are constants.

Equations (1) and (2) are solved by a simple iterative procedure suggested by Huber:

$$\hat{\sigma}_k = \sqrt{\frac{1}{\beta(n-1)}\left(\sum_{i=1}^{n}\chi\left(\frac{x_i-\hat{\theta}_{k-1}}{\hat{\sigma}_{k-1}}\right)\right)\hat{\sigma}_{k-1}^2}$$

and

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \frac{1}{n}\sum_{i=1}^{n}\psi\left(\frac{x_i-\hat{\theta}_{k-1}}{\hat{\sigma}_k}\right)\hat{\sigma}_k$$

or

$\hat{\sigma}_k = \sigma_c$ if σ is fixed.

The initial values for $\hat{\theta}$ and $\hat{\sigma}$ may either be user-supplied or calculated within G07DBF as the sample median and an estimate of σ based on the median absolute deviation respectively.

G07DBF is based upon subroutine LYHALG within the ROBETH library, see Marazzi [3].

4. References

[1] HAMPEL, F.R., RONCHETTI, E.M., ROUSSEEUW, P.J. and STAHEL, W.A.
Robust Statistics. The Approach Based on Influence Functions.
Wiley, 1986.

[2] HUBER, P.J.
Robust Statistics.
Wiley, 1981.

[3] MARAZZI, A.
Subroutines for Robust Estimation of Location and Scale in ROBETH.
Institut Universitaire de Médecine Sociale et Préventive, Lausanne, 1987.
(Cah Rech Doc IUMSP, No 3 ROB 1).

5. Parameters

1: ISIGMA – INTEGER. *Input*

On entry: the value assigned to ISIGMA determines whether $\hat{\sigma}$ is to be simultaneously estimated.

ISIGMA = 0

The estimation of $\hat{\sigma}$ is bypassed and SIGMA is set equal to σ_c;

ISIGMA = 1

$\hat{\sigma}$ is estimated simultaneously.

2: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N > 1.

3: X(N) – ***real*** array. *Input*

On entry: the vector of observations, $x_1,x_2,...,x_n$.

4: IPSI – INTEGER. *Input*

On entry: which ψ function is to be used.

IPSI = 0,

$\psi(t) = t$

IPSI = 1,

Huber's function,

IPSI = 2,

Hampel's piecewise linear function,

IPSI = 3,

Andrew's sine wave,

IPSI = 4,

Tukey's bi-weight.

5: C – ***real.*** *Workspace*

If IPSI = 1 on entry, C must specify the parameter, c, of Huber's ψ function. C is not referenced if IPSI ≠ 1.

Constraint: C > 0.0 if IPSI = 1.

6: H1 – ***real.*** *Input*
7: H2 – ***real.*** *Input*
8: H3 – ***real.*** *Input*

If IPSI = 2 on entry, H1, H2, and H3 must specify the parameters h_1, h_2, and h_3, of Hampel's piecewise linear ψ function. H1, H2, and H3 are not referenced if IPSI ≠ 2.

Constraint: 0 ≤ H1 ≤ H2 ≤ H3 and H3 > 0.0 if IPSI = 2.

9: DCHI – ***real.*** *Input*

On entry: the parameter, d, of the χ function. DCHI is not referenced if IPSI = 0.

Constraint: DCHI > 0.0 if IPSI ≠ 0.

10: THETA – ***real.*** *Input/Output*

On entry: if SIGMA > 0 then THETA must be set to the required starting value of the estimation of the location parameter $\hat{\theta}$. A reasonable initial value for $\hat{\theta}$ will often be the sample mean or median.

On exit: the M-estimate of the location parameter, $\hat{\theta}$.

11: SIGMA – ***real.*** *Input/Output*

The role of SIGMA depends on the value assigned to ISIGMA (see above) as follows:

ISIGMA = 1

On entry: SIGMA must be assigned a value which determines the values of the starting points for the calculations of $\hat{\theta}$ and $\hat{\sigma}$. If SIGMA ≤ 0.0 then G07DBF will determine the starting points of $\hat{\theta}$ and $\hat{\sigma}$. Otherwise the value assigned to SIGMA will be taken as the starting point for $\hat{\sigma}$, and THETA must be assigned a value before entry, see above.

ISIGMA = 0

On entry: SIGMA must be assigned a value which determines the value of σ_c, which is held fixed during the iterations, and the starting value for the calculation of $\hat{\theta}$. If SIGMA $\le$ 0, then G07DBF will determine the value of σ_c as the median absolute deviation adjusted to reduce bias (see G07DAF) and the starting point for $\hat{\theta}$. Otherwise, the value assigned to SIGMA will be taken as the value of σ_c and THETA must be assigned a relevant value before entry, see above.

On exit: SIGMA contains the M-estimate of the scale parameter, $\hat{\sigma}$, if ISIGMA was assigned the value 1 on entry, otherwise SIGMA will contain the initial fixed value σ_c.

12: MAXIT – INTEGER. *Input*

On entry: the maximum number of iterations that should be used during the estimation.

Suggested value: MAXIT = 50.

Constraint: MAXIT > 0.

13: TOL – ***real.*** *Input*

On entry: the relative precision for the final estimates. Convergence is assumed when the increments for THETA, and SIGMA are less than TOL$\times$max$(1.0,\sigma_{k-1})$.

Constraint: TOL > 0.0.

14: RS(N) – ***real*** array. *Output*

On exit: the Winsorized residuals.

15: NIT – INTEGER. *Output*

On exit: the number of iterations that were used during the estimation.

16: WRK(N) – ***real*** array. *Output*

On exit: if SIGMA $\le$ 0.0 on entry, WRK will contain the n observations in ascending order.

17: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, N $\le$ 1,
or MAXIT $\le$ 0,
or TOL $\le$ 0.0,
or ISIGMA $\ne$ 0 or 1,
or IPSI < 0,
or IPSI > 4.

IFAIL = 2

On entry, C $\le$ 0.0 and IPSI = 1,
or H1 < 0.0 and IPSI = 2,
or H1 = H2 = H3 = 0.0 and IPSI = 2,
or H1 > H2 and IPSI = 2,

or H1 > H3 and IPSI = 2,
or H2 > H3 and IPSI = 2,
or DCHI $\leq$ 0.0 and IPSI $\neq$ 0.

IFAIL = 3

On entry, all elements of the input array X are equal.

IFAIL = 4

SIGMA, the current estimate of σ, is zero or negative. This error exit is very unlikely, although it may be caused by too large an initial value of SIGMA.

IFAIL = 5

The number of iterations required exceeds MAXIT.

IFAIL = 6

On completion of the iterations, the Winsorized residuals were all zero. This may occur when using the ISIGMA = 0 option with a redescending ψ function, i.e. Hampel's piecewise linear function, Andrew's sine wave, and Tukey's biweight.

If the given value of σ is too small, then the standardised residuals $\frac{x_i - \hat{\theta}_k}{\sigma_c}$, will be large and all the residuals may fall into the region for which $\psi(t) = 0$. This may incorrectly terminate the iterations thus making THETA and SIGMA invalid.

Re-enter the routine with a larger value of σ_c or with ISIGMA = 1.

7. Accuracy

On successful exit the accuracy of the results is related to the value of TOL, see Section 5.

8. Further Comments

When the user supplies the initial values, care has to be taken over the choice of the initial value of σ. If too small a value of σ is chosen then initial values of the standardized residuals $\frac{x_i - \hat{\theta}_k}{\sigma}$ will be large. If the redescending ψ functions are used, i.e. Hampel's piecewise linear function, Andrew's sine wave, or Tukey's bi-weight, then these large values of the standardised residuals are Winsorized as zero. If a sufficient number of the residuals fall into this category then a false solution may be returned, see Hampel [1] page 152.

9. Example

The following program reads in a set of data consisting of eleven observations of a variable X.

For this example, Hampels's Piecewise Linear Function is used (IPSI = 2), values for h_1, h_2 and h_3 along with d for the χ function, being read from the data file.

Using the following starting values various estimates of θ and σ are calculated and printed along with the number of iterations used:

(a) G07DBF determines the starting values, σ is estimated simultaneously.
(b) The user supplies the starting values, σ is estimated simultaneously.
(c) G07DBF determines the starting values, σ is fixed.
(d) The user supplies the starting values, σ is fixed.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07DBF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=25)
*     .. Local Scalars ..
      real             C, DCHI, H1, H2, H3, SIGMA, SIGSAV, THESAV,
     +                 THETA, TOL
      INTEGER          I, IFAIL, IPSI, ISIGMA, MAXIT, N, NIT
*     .. Local Arrays ..
      real             RS(NMAX), WRK(NMAX), X(NMAX)
*     .. External Subroutines ..
      EXTERNAL         G07DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07DBF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      WRITE (NOUT,*)
      IF (N.LE.NMAX) THEN
         READ (NIN,*) (X(I),I=1,N)
         READ (NIN,*) IPSI, H1, H2, H3, DCHI, MAXIT
         WRITE (NOUT,*)
     +     '           Input parameters     Output parameters'
         WRITE (NOUT,*) 'ISIGMA    SIGMA    THETA    TOL      SIGMA    THETA'
   20    READ (NIN,*,END=40) ISIGMA, SIGMA, THETA, TOL
         SIGSAV = SIGMA
         THESAV = THETA
         IFAIL = 0
*
         CALL G07DBF(ISIGMA,N,X,IPSI,C,H1,H2,H3,DCHI,THETA,SIGMA,MAXIT,
     +               TOL,RS,NIT,WRK,IFAIL)
*
         WRITE (NOUT,99999) ISIGMA, SIGSAV, THESAV, TOL, SIGMA, THETA
         GO TO 20
      ELSE
         WRITE (NOUT,99998) 'N is out of range: N =', N
      END IF
   40 STOP
*
99999 FORMAT (1X,I3,3X,2F8.4,F7.4,F9.4,F8.4,I4)
99998 FORMAT (1X,A,I5)
      END
```

9.2. Program Data

```
G07DBF Example Program Data
11                                         : NUMBER OF OBSERVATIONS
13.0 11.0 16.0 5.0 3.0 18.0 9.0 8.0 6.0 27.0 7.0  : OBSERVATIONS
 2    1.5  3.0 4.5  1.5  50                :IPSI  H1   H2   H3  DCHI  MAXIT
  1    -1.0    0.0   0.0001                :ISIGMA  SIGMA  THETA   TOL
  1     7.0    2.0   0.0001
  0    -1.0    0.0   0.0001
  0     7.0    2.0   0.0001
```

9.3. Program Results

```
G07DBF Example Program Results

           Input  parameters        Output  parameters
ISIGMA    SIGMA    THETA    TOL      SIGMA   THETA
  1     -1.0000  0.0000 0.0001     6.3247  10.5487
  1      7.0000  2.0000 0.0001     6.3249  10.5487
  0     -1.0000  0.0000 0.0001     5.9304  10.4896
  0      7.0000  2.0000 0.0001     7.0000  10.6500
```

G07DCF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07DCF computes a M-estimate of location with (optional) simultaneous estimation of scale, where the user provides the weight functions.

2. Specification

```
 SUBROUTINE G07DCF (CHI, PSI, ISIGMA, N, X, BETA, THETA, SIGMA,
1                   MAXIT, TOL, RS, NIT, WRK, IFAIL)
 INTEGER          ISIGMA, N, MAXIT, NIT, IFAIL
 real             X(N), BETA, THETA, SIGMA, TOL, RS(N),
1                 WRK(N)
 EXTERNAL         CHI, PSI
```

3. Description

The data consists of a sample of size n, denoted by $x_1,x_2,...,x_n$, drawn from a random variable X.

The x_i are assumed to be independent with an unknown distribution function of the form,

$$F((x_i-\theta)/\sigma)$$

where θ is a location parameter, and σ is a scale parameter. M-estimators of θ and σ are given by the solution to the following system of equations;

$$\sum_{i=1}^{n} \psi((x_i-\hat{\theta})/\hat{\sigma}) = 0$$

$$\sum_{i=1}^{n} \chi((x_i-\hat{\theta})/\hat{\sigma}) = (n-1)\beta$$

where ψ and χ are user-supplied weight functions, and β is a constant. Optionally the second equation can be omitted and the first equation is solved for $\hat{\theta}$ using an assigned value of $\sigma = \sigma_c$.

The constant β should be chosen so that $\hat{\sigma}$ is an unbiased estimator when x_i, for $i = 1,2,...n$ has a normal distribution. To achieve this the value of β is calculated as:

$$\beta = E(\chi) = \int_{-\infty}^{\infty} \chi(z)\frac{1}{\sqrt{2\pi}} \exp\left\{\frac{-z^2}{2}\right\}dz$$

The values of $\psi\left(\frac{x_i-\hat{\theta}}{\hat{\sigma}}\right)\hat{\sigma}$ are known as the Winsorized residuals.

The equations are solved by a simple iterative procedure, suggested by Huber:

$$\hat{\sigma}_k = \sqrt{\frac{1}{\beta(n-1)}\left(\sum_{i=1}^{n}\chi\left(\frac{x_i-\hat{\theta}_{k-1}}{\hat{\sigma}_{k-1}}\right)\right)\hat{\sigma}_{k-1}^2}$$

and

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \frac{1}{n}\sum_{i=1}^{n}\psi\left(\frac{x_i-\hat{\theta}_{k-1}}{\hat{\sigma}_k}\right)\hat{\sigma}_k$$

or

$$\hat{\sigma}_k = \sigma_c$$

if σ is fixed.

The initial values for $\hat{\theta}$ and $\hat{\sigma}$ may be user-supplied or calculated within G07DBF as the sample median and an estimate of σ based on the median absolute deviation respectively.

G07DCF is based upon subroutine LYHALG within the ROBETH library, see Marazzi [3].

4. References

[1] HAMPEL, F.R., RONCHETTI, E.M., ROUSSEEUW, P.J. and STAHEL, W.A.
Robust Statistics. The Approach Based on Influence Functions.
Wiley 1986.

[2] HUBER, P.J.
Robust Statistics.
Wiley 1981.

[3] MARAZZI, A.
Subroutines for Robust Estimation of Location and Scale in ROBETH.
Institut Universitaire de Médecine Sociale et Préventive, Lausanne, 1987.
(Cah Rech Doc IUMSP, No 3 ROB 1).

5. Parameters

1: CHI – ***real*** FUNCTION, supplied by the user. *External Procedure*

CHI must return the value of the weight function χ for a given value of its argument. The value of χ must be non-negative.

Its specification is:

```
real FUNCTION CHI(T)
real          T
```

1: T – ***real.*** *Input*

On entry: the argument for which CHI must be evaluated.

CHI must be declared as EXTERNAL in the (sub)program from which G07DCF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: PSI – ***real*** FUNCTION, supplied by the user. *External Procedure*

PSI must return the value of the weight function ψ for a given value of its argument.

Its specification is:

```
real FUNCTION PSI(T)
real          T
```

1: T – ***real.*** *Input*

On entry: the argument for which PSI must be evaluated.

PSI must be declared as EXTERNAL in the (sub)program from which G07DCF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

3: ISIGMA – INTEGER. *Input*

On entry: the value assigned to ISIGMA determines whether $\hat{\sigma}$ is to be simultaneously estimated.

ISIGMA = 0

The estimation of $\hat{\sigma}$ is bypassed and SIGMA is set equal to σ_c;

ISIGMA = 1

$\hat{\sigma}$ is estimated simultaneously.

4: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N > 1.

5: X(N) – ***real*** array. *Input*

On entry: the vector of observations, $x_1,x_2,...,x_n$.

6: BETA – ***real***. *Input*

On entry: the value of the constant β of the chosen CHI function.

Constraint: BETA > 0.0.

7: THETA – ***real***. *Input/Output*

On entry: if SIGMA > 0, then THETA must be set to the required starting value of the estimate of the location parameter $\hat{\theta}$. A reasonable initial value for $\hat{\theta}$ will often be the sample mean or median.

On exit: the M-estimate of the location parameter $\hat{\theta}$.

8: SIGMA – ***real***. *Input/Output*

The role of SIGMA depends on the value assigned to ISIGMA (see above) as follows:

ISIGMA = 1

On entry: SIGMA must be assigned a value which determines the values of the starting points for the calculation of $\hat{\theta}$ and $\hat{\sigma}$. If SIGMA ≤ 0.0, then G07DCF will determine the starting points of $\hat{\theta}$ and $\hat{\sigma}$. Otherwise, the value assigned to SIGMA will be taken as the starting point for $\hat{\sigma}$, and THETA must be assigned a relevant value before entry, see above.

ISIGMA = 0

On entry: SIGMA must be assigned a value which determines the values of σ_c, which is held fixed during the iterations, and the starting value for the calculation of $\hat{\theta}$. If SIGMA ≤ 0, then G07DCF will determine the value of σ_c as the median absolute deviation adjusted to reduce bias (see G07DAF) and the starting point for θ. Otherwise, the value assigned to SIGMA will be taken as the value of σ_c and THETA must be assigned a relevant value before entry, see above.

On exit: the M-estimate of the scale parameter $\hat{\sigma}$, if ISIGMA was assigned the value 1 on entry, otherwise SIGMA will contain the initial fixed value σ_c.

9: MAXIT – INTEGER. *Input*

On entry: the maximum number of iterations that should be used during the estimation.

Suggested value: MAXIT = 50.

Constraint: MAXIT > 0.

10: TOL – ***real***. *Input*

On entry: the relative precision for the final estimates. Convergence is assumed when the increments for THETA, and SIGMA are less than TOL×max$(1.0,\sigma_{k-1})$.

Constraint: TOL > 0.0.

11: RS(N) – ***real*** array. *Output*

On exit: the Winsorized residuals.

12: NIT – INTEGER. *Output*

On exit: the number of iterations that were used during the estimation.

13: WRK(N) – ***real*** array. *Output*

On exit: if SIGMA $\leq$ 0.0 on entry, WRK will contain the n observations in ascending order.

14: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, N $\leq$ 1,
or MAXIT $\leq$ 0,
or TOL $\leq$ 0.0,
or ISIGMA $\neq$ 0 or 1.

IFAIL = 2

On entry, BETA $\leq$ 0.0.

IFAIL = 3

On entry, all elements of the input array X are equal.

IFAIL = 4

SIGMA, the current estimate of σ, is zero or negative. This error exit is very unlikely, although it may be caused by too large an initial value of SIGMA.

IFAIL = 5

The number of iterations required exceeds MAXIT.

IFAIL = 6

On completion of the iterations, the Winsorized residuals were all zero. This may occur when using the ISIGMA = 0 option with a redescending ψ function, i.e. $\psi = 0$ if $|t| > \tau$, for some positive constant τ.

If the given value of σ is too small, then the standardized residuals $\dfrac{x_i - \hat{\theta}_k}{\sigma_c}$, will be large and all the residuals may fall into the region for which $\psi(t) = 0$. This may incorrectly terminate the iterations thus making THETA and SIGMA invalid.

Re-enter the routine with a larger value of σ_c or with ISIGMA = 1.

IFAIL = 7

The value returned by the CHI function is negative.

7. Accuracy

On successful exit the accuracy of the results is related to the value of TOL, see Section 5.

8. Further Comments

Standard forms of the functions ψ and χ are given in Hampel *et al.* [1], Huber [2], and Marazzi [3]. G07DBF calculates M-estimates using some standard forms for ψ and χ.

When the user supplies the initial values, care has to be taken over the choice of the initial value of σ. If too small a value is chosen then initial values of the standardized residuals $\frac{x_i - \hat{\theta}_k}{\sigma}$ will be large. If the redescending ψ functions are used, i.e. $\psi = 0$ if $|t| > \tau$, for some positive constant τ, then these large values are Winsorized as zero. If a sufficient number of the residuals fall into this category then a false solution may be returned, see Hampel *et al.* [1] page 152.

9. Example

The following program reads in a set of data consisting of eleven observations of a variable X.

The PSI and CHI functions used are Hampel's Piecewise Linear Function and Hubers CHI function respectively.

Using the following starting values various estimates of θ and σ are calculated and printed along with the number of iterations used:

(a) G07DCF determined the starting values, σ is estimated simultaneously.
(b) The user supplies the starting values, σ is estimated simultaneously.
(c) G07DCF determined the starting values, σ is fixed.
(d) The user supplies the starting values, σ is fixed.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07DCF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=25)
*     .. Local Scalars ..
      real             BETA, SIGMA, SIGSAV, THESAV, THETA, TOL
      INTEGER          I, IFAIL, ISIGMA, MAXIT, N, NIT
*     .. Local Arrays ..
      real             RS(NMAX), WRK(NMAX), X(NMAX)
*     .. External Functions ..
      real             CHI, PSI
      EXTERNAL         CHI, PSI
*     .. External Subroutines ..
      EXTERNAL         G07DCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07DCF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      WRITE (NOUT,*)
      IF (N.LE.NMAX) THEN
         READ (NIN,*) (X(I),I=1,N)
         READ (NIN,*) BETA, MAXIT
         WRITE (NOUT,*)
     +     '             Input parameters     Output parameters'
         WRITE (NOUT,*) 'ISIGMA   SIGMA   THETA   TOL     SIGMA  THETA'
   20    READ (NIN,*,END=40) ISIGMA, SIGMA, THETA, TOL
         SIGSAV = SIGMA
         THESAV = THETA
         IFAIL = 0
*
```

```
         CALL G07DCF(CHI,PSI,ISIGMA,N,X,BETA,THETA,SIGMA,MAXIT,TOL,RS,
     +               NIT,WRK,IFAIL)
*
         WRITE (NOUT,99999) ISIGMA, SIGSAV, THESAV, TOL, SIGMA, THETA
         GO TO 20
      ELSE
         WRITE (NOUT,99998) 'N is out of range: N =', N
      END IF
   40 STOP
*
99999 FORMAT (1X,I3,3X,2F8.4,F7.4,1X,2F8.4)
99998 FORMAT (1X,A,I5)
      END
*
      real FUNCTION PSI(T)
*     Hampel's Piecewise Linear Function.
*     .. Parameters ..
      real              ZERO
      PARAMETER         (ZERO=0.0e+0)
      real              H1, H2, H3
      PARAMETER         (H1=1.5e0,H2=3.0e0,H3=4.5e0)
*     .. Scalar Arguments ..
      real              T
*     .. Local Scalars ..
      real              ABST
*     .. Intrinsic Functions ..
      INTRINSIC         ABS, MIN
*     .. Executable Statements ..
      ABST = ABS(T)
      IF (ABST.LT.H3) THEN
         IF (ABST.LE.H2) THEN
            PSI = MIN(H1,ABST)
         ELSE
            PSI = H1*(H3-ABST)/(H3-H2)
         END IF
         IF (T.LT.ZERO) PSI = -PSI
      ELSE
         PSI = ZERO
      END IF
      RETURN
      END
*
      real FUNCTION CHI(T)
*     Huber's CHI function.
*     .. Parameters ..
      real              DCHI
      PARAMETER         (DCHI=1.5e0)
*     .. Scalar Arguments ..
      real              T
*     .. Local Scalars ..
      real              ABST, PS
*     .. Intrinsic Functions ..
      INTRINSIC         ABS, MIN
*     .. Executable Statements ..
      ABST = ABS(T)
      PS = MIN(DCHI,ABST)
      CHI = PS*PS/2
      RETURN
      END
```

9.2. Program Data

```
G07DCF Example Program Data
11                                : N, NUMBER OF OBSERVATIONS
13.0 11.0 16.0 5.0 3.0 18.0 9.0 8.0 6.0 27.0 7.0  : X, OBSERVATIONS
0.3892326     50                  : BETA    MAXIT
  1     -1.0    0.0   0.0001      : ISIGMA  SIGMA  THETA   TOL
  1      7.0    2.0   0.0001
  0     -1.0    0.0   0.0001
  0      7.0    2.0   0.0001
```

9.3. Program Results

```
 G07DCF Example Program Results

              Input  parameters       Output  parameters
ISIGMA     SIGMA    THETA    TOL      SIGMA   THETA
   1     -1.0000   0.0000  0.0001    6.3247  10.5487
   1      7.0000   2.0000  0.0001    6.3249  10.5487
   0     -1.0000   0.0000  0.0001    5.9304  10.4896
   0      7.0000   2.0000  0.0001    7.0000  10.6500
```

G07DDF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07DDF calculates the trimmed and Winsorized means of a sample and estimates of the variances of the two means.

2. Specification

```
      SUBROUTINE G07DDF (N, X, ALPHA, TMEAN, WMEAN, TVAR, WVAR, K,
     1                   SX, IFAIL)
      INTEGER            N, K, IFAIL
      real               X(N), ALPHA, TMEAN, WMEAN, TVAR, WVAR, SX(N)
```

3. Description

G07DDF calculates the α-trimmed mean and α-Winsorized mean for a given α, as described below.

Let x_i, for $i = 1,2,...,n$ represent the n sample observations sorted into ascending order. Let $k = [\alpha n]$ where $[y]$ represents the integer part of y.

Then the trimmed mean is defined as;

$$\bar{x}_t = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} x_i,$$

and the winsorized mean is defined as;

$$\bar{x}_w = \frac{1}{n} \sum_{i=k+1}^{n-k} x_i + (kx_{k+1}) + (kx_{n-k}).$$

G07DDF then calculates the Winsorized variance about the trimmed and Winsorized means respectively and divides by n to obtain estimates of the variances of the above two means.

Thus we have;

$$\text{Estimate of var}(\bar{x}_t) = \frac{1}{n^2} \sum_{i=k+1}^{n-k} (x_i-\bar{x}_t)^2 + k(x_{k+1}-\bar{x}_t)^2 + k(x_{n-k}-\bar{x}_t)^2$$

and

$$\text{Estimate of var}(\bar{x}_w) = \frac{1}{n^2} \sum_{i=k+1}^{n-k} (x_i-\bar{x}_w)^2 + k(x_{k+1}-\bar{x}_w)^2 + k(x_{n-k}-\bar{x}_w)^2.$$

4. References

[1] HUBER, P.J.
Robust Statistics.
Wiley, New York, 1981.

[2] HAMPEL, F.R., RONCHETTI, E.M., ROUSSEEUW, P.J., STAHEL, W.A.
Robust Statistics, The approach based on Influence Functions.
Wiley, New York, 1986.

5. Parameters

1: N – INTEGER. *Input*

On entry: the number of observations, n.

Constraint: N $\geq$ 2.

2: X(N) – ***real*** array. *Input*

On entry: the sample observations, x_i, for $i = 1,2,...,n$.

3: ALPHA – ***real.*** *Input*

On entry: the proportion of observations to be trimmed at each end of the sorted sample, α.

Constraint: $0.0 \le \text{ALPHA} < 0.5$.

4: TMEAN – ***real.*** *Output*

On exit: the α-trimmed mean, $\bar{x}_t$.

5: WMEAN – ***real.*** *Output*

On exit: the α-Winsorized mean, $\bar{x}_w$.

6: TVAR – ***real.*** *Output*

On exit: contains an estimate of the variance of the trimmed mean.

7: WVAR – ***real.*** *Output*

On exit: contains an estimate of the variance of the Winsorized mean.

8: K – INTEGER. *Output*

On exit: contains the number of observations trimmed at each end, k.

9: SX(N) – ***real*** array. *Output*

On exit: contains the sample observations sorted into ascending order.

10: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, $N \le 1$.

IFAIL = 2

On entry, ALPHA < 0.0,
or ALPHA ≥ 0.5.

7. Accuracy

The results should be accurate to within a small multiple of ***machine precision.***

8. Further Comments

The time taken by the routine is proportional to n.

9. Example

The following program finds the α-trimmed mean and α-Winsorized mean for a sample of 16 observations where $\alpha = 0.15$. The estimates of the variances of the above two means are also calculated.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07DDF Example Program Text
*     Mark 14 Release.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=1000)
*     .. Local Scalars ..
      real             ALPHA, PROPN, TMEAN, TVAR, WMEAN, WVAR
      INTEGER          I, IFAIL, K, N
*     .. Local Arrays ..
      real             SX(NMAX), X(NMAX)
*     .. External Subroutines ..
      EXTERNAL         G07DDF
*     .. Intrinsic Functions ..
      INTRINSIC        real
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07DDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, (X(I),I=1,N), ALPHA
      IFAIL = 0
*
      CALL G07DDF(N,X,ALPHA,TMEAN,WMEAN,TVAR,WVAR,K,SX,IFAIL)
*
      PROPN = real(K)/N
      PROPN = 100.0e0 - 200.0e0*PROPN
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'Statistics from middle ', PROPN, '% of data'
      WRITE (NOUT,*)
      WRITE (NOUT,99998) '                Trimmed-mean = ', TMEAN
      WRITE (NOUT,99998) '    Variance of Trimmed-mean = ', TVAR
      WRITE (NOUT,*)
      WRITE (NOUT,99998) '             Winsorized-mean = ', WMEAN
      WRITE (NOUT,99998) 'Variance of Winsorized-mean = ', WVAR
      STOP
*
99999 FORMAT (1X,A,F6.2,A)
99998 FORMAT (1X,A,F11.4)
      END
```

9.2. Program Data

```
G07DDF Example Program Data
 16
 26.0 12.0 9.0 2.0 5.0 6.0 8.0 14.0 7.0 3.0 1.0 11.0 10.0 4.0 17.0 21.0
 0.15
```

9.3. Program Results

```
 G07DDF Example Program Results

 Statistics from middle  75.00% of data

                Trimmed-mean =      8.8333
    Variance of Trimmed-mean =      1.5434

             Winsorized-mean =      9.1250
 Variance of Winsorized-mean =      1.5381
```

G07EAF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07EAF computes a rank based (nonparametric) estimate and confidence interval for the location parameter of a single population.

2. Specification

```
      SUBROUTINE G07EAF (METHOD, N, X, CLEVEL, THETA, THETAL, THETAU, ESTCL,
     1                   WLOWER, WUPPER, WRK, IWRK, IFAIL)
      INTEGER            N, IWRK(3*N), IFAIL
      real               X(N), CLEVEL, THETA, THETAL, THETAU, ESTCL, WLOWER,
     1                   WUPPER, WRK(4*N)
      CHARACTER*1        METHOD
```

3. Description

Consider a vector of independent observations, $x = (x_1,x_2,...,x_n)^T$ with unknown common symmetric density $f(x_i-\theta)$. G07EAF computes the Hodges-Lehmann location estimator (see Lehmann [1]) of the centre of symmetry θ, together with an associated confidence interval. The Hodges-Lehmann estimate is defined as

$$\hat{\theta} = \text{median}\left\{\frac{x_i + x_j}{2},\ 1 \le i \le j \le n\right\}.$$

Let $m = \frac{n(n+1)}{2}$ and let a_k, for $k = 1,2,...,m$ denote the m ordered averages $\frac{x_i + x_j}{2}$ for $1 \le i \le j \le n$. Then

$$\text{if } m \text{ is odd, } \hat{\theta} = a_k \text{ where } k = \frac{(m+1)}{2},$$

$$\text{if } m \text{ is even, } \hat{\theta} = \frac{a_k + a_{k+1}}{2} \text{ where } k = \frac{m}{2}.$$

This estimator arises from inverting the one-sample Wilcoxon signed-rank test statistic, $W(x-\theta_0)$, for testing the hypothesis that $\theta = \theta_0$. Effectively $W(x-\theta_0)$ is a monotonically decreasing step function of θ_0 with

$$\text{mean}(W) = \mu = \frac{n(n+1)}{4},$$

$$\text{var}(W) = \sigma^2 = \frac{n(n+1)(2n+1)}{24}.$$

The estimate $\hat{\theta}$ is the solution to the equation $W(x-\hat{\theta}) = \mu$; two methods are available for solving this equation. These methods avoid the computation of all the ordered averages a_k; this is because for large n both the storage requirements and the computation time would be excessive.

The first is an exact method based on a set partitioning procedure on the set of all ordered averages $(x_i+x_j)/2$ for $i \le j$. This is based on the algorithm proposed by Monahan [4].

The second is an iterative algorithm, based on the Illinois method which is a modification of the regula falsi method, see McKean and Ryan [3]. This algorithm has proved suitable for the function $W(x-\theta_0)$ which is asymptotically linear as a function of θ_0.

The confidence interval limits are also based on the inversion of the Wilcoxon test statistic.

Given a desired percentage for the confidence interval, $1-\alpha$, expressed as a proportion between 0 and 1, initial estimates for the lower and upper confidence limits of the Wilcoxon statistic are found from

$$W_l = \mu - 0.5 + (\sigma\Phi^{-1}(\alpha/2)) \text{ and}$$
$$W_u = \mu + 0.5 + (\sigma\Phi^{-1}(1-\alpha/2)),$$

where Φ^{-1} is the inverse cumulative Normal distribution function.

W_l and W_u are rounded to the nearest integer values. These estimates are then refined using an exact method if $n \leq 80$, and a Normal approximation otherwise, to find W_l and W_u satisfying

$$P(W \leq W_l) \leq \alpha/2$$
$$P(W \leq W_l + 1) > \alpha/2$$

and

$$P(W \geq W_u) \leq \alpha/2$$
$$P(W \geq W_u - 1) > \alpha/2.$$

Let $W_u = m - k$; then $\theta_l = a_{k+1}$. This is the largest value θ_l such that $W(x-\theta_l) = W_u$.

Let $W_l = k$; then $\theta_u = a_{m-k}$. This is the smallest value θ_u such that $W(x-\theta_u) = W_l$.

As in the case of $\hat{\theta}$, these equations may be solved using either the exact and iterative methods to find the values θ_l and θ_u.

Then (θ_l,θ_u) is the confidence interval for θ. The confidence interval is thus defined by those values of θ_0 such that the null hypothesis, $\theta = \theta_0$, is not rejected by the Wilcoxon signed-rank test at the $(100\times\alpha)\%$ level.

4. References

[1] LEHMANN, E.L.
Nonparametrics. Statistical Methods Based on Ranks.
McGraw-Hill, New York, 1975.

[2] MARAZZI, A.
Subroutines for robust estimation of location and scale in ROBETH, ROBETH-85 Document No. 1, Aug. 1985.
Institut Universitaire de Medecine Sociale et Préventive, Lausanne, 1985.

[3] McKEAN, J.W. and RYAN, T.A.
Algorithm 516: An algorithm for obtaining confidence intervals and point estimates based on ranks in the two-sample location problem.
ACM Trans. Math. Softw., 3, pp. 183-185, 1977.

[4] MONAHAN, J.F.
Algorithm 616: Fast computation of the Hodges-Lehmann location estimator.
ACM Trans. Math. Softw., 10, pp. 265-270, 1984.

5. Parameters

1: METHOD – CHARACTER*1. *Input*

On entry: specifies the method to be used;

if METHOD = 'E' the exact algorithm is used,
if METHOD = 'A' the iterative algorithm is used.

Constraint: METHOD = 'E' or 'A'.

2: N – INTEGER. *Input*

On entry: the sample size, n.

Constraint: N $\geq$ 2.

3: X(N) – ***real*** array. *Input*

On entry: the sample observations, x_i for $i = 1,2,...,n$.

4: CLEVEL – ***real.*** *Input*

On entry: the confidence interval desired.

e.g. for a 95% confidence interval set CLEVEL = 0.95.

Constraint: 0.0 < CLEVEL < 1.0.

5: THETA – ***real.*** *Output*

On exit: the estimate of the location, $\hat{\theta}$.

6: THETAL – ***real.*** *Output*

On exit: the estimate of the lower limit of the confidence interval, θ_l.

7: THETAU – ***real.*** *Output*

On exit: the estimate of the upper limit of the confidence interval, θ_u.

8: ESTCL – ***real.*** *Output*

On exit: an estimate of the actual percentage confidence of the interval found, as a proportion between (0.0,1.0).

9: WLOWER – ***real.*** *Output*

On exit: the upper value of the Wilcoxon test statistic, W_u, corresponding to the lower limit of the confidence interval.

10: WUPPER – ***real.*** *Output*

On exit: the lower value of the Wilcoxon test statistic, W_l, corresponding to the upper limit of the confidence interval.

11: WRK(4*N) – ***real*** array. *Workspace*

12: IWRK(3*N) – INTEGER array. *Workspace*

13: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, METHOD ≠ 'E' or 'A',
or N < 2,
or CLEVEL ≤ 0.0,
or CLEVEL ≥ 1.0.

IFAIL = 2

There is not enough information to compute a confidence interval since the whole sample consists of identical values.

IFAIL = 3

For at least one of the estimates $\hat{\theta}$, θ_l and θ_u, the underlying iterative algorithm (when METHOD = 'A') failed to converge. This is an unlikely exit but the estimate should still be a reasonable approximation.

7. Accuracy

The routine should produce results accurate to 5 significant figures in the width of the confidence interval; that is the error for any one of the three estimates should be less than $0.00001\times$(THETAU–THETAL).

8. Further Comments

The time taken increases with the sample size n.

9. Example

The following program calculates a 95% confidence interval for θ, a measure of symmetry of the sample of 50 observations.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07EAF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=100)
*     .. Local Scalars ..
      real             CLEVEL, ESTCL, THETA, THETAL, THETAU, WLOWER,
     +                 WUPPER
      INTEGER          I, IFAIL, N
*     .. Local Arrays ..
      real             WRK(4*NMAX), X(NMAX)
      INTEGER          IWRK(3*NMAX)
*     .. External Subroutines ..
      EXTERNAL         G07EAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07EAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.1 .OR. N.GT.NMAX) THEN
         WRITE (NOUT,99999) N
      ELSE
         READ (NIN,*) (X(I),I=1,N)
         READ (NIN,*) CLEVEL
         IFAIL = 0
*
         CALL G07EAF('Exact',N,X,CLEVEL,THETA,THETAL,THETAU,ESTCL,
     +               WLOWER,WUPPER,WRK,IWRK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Location estimator     Confidence Interval '
         WRITE (NOUT,*)
         WRITE (NOUT,99998) THETA, '( ', THETAL, ' , ', THETAU, ' )'
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Corresponding Wilcoxon statistics'
         WRITE (NOUT,*)
         WRITE (NOUT,99997) ' Lower : ', WLOWER
         WRITE (NOUT,99997) ' Upper : ', WUPPER
      END IF
      STOP
```

```
*
99999 FORMAT (1X,'N is less than 2 or greater than NMAX : N = ',I8)
99998 FORMAT (3X,F10.4,12X,A,F6.4,A,F6.4,A)
99997 FORMAT (A,F8.2)
      END
```

9.2. Program Data

```
G07EAF Example Program Data
 40
 -0.23   0.35  -0.77   0.35   0.27   -0.72   0.08  -0.40  -0.76   0.45
  0.73   0.74   0.83  -0.87   0.21    0.29  -0.91  -0.04   0.82  -0.38
 -0.31   0.24  -0.47  -0.68  -0.77   -0.86  -0.59   0.73   0.39  -0.44
  0.63  -0.22  -0.07  -0.43  -0.21   -0.31   0.64  -1.00  -0.86  -0.73
  0.95
```

9.3. Program Results

```
 G07EAF Example Program Results

  Location estimator      Confidence Interval

      -0.1300             ( -.3300 , 0.0350 )

  Corresponding Wilcoxon statistics

 Lower :   556.00
 Upper :   264.00
```

G07EBF – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

G07EBF calculates a rank based (nonparametric) estimate and confidence interval for the difference in location between two independent populations.

2. Specification

```
      SUBROUTINE G07EBF (METHOD, N, X, M, Y, CLEVEL, THETA, THETAL, THETAU,
     1                   ESTCL, ULOWER, UUPPER, WRK, IWRK, IFAIL)
      INTEGER          N, M, IWRK(3*N), IFAIL
      real             X(N), Y(M), CLEVEL, THETA, THETAL, THETAU, ESTCL,
     1                 ULOWER, UUPPER, WRK(3*(M+N))
      CHARACTER*1      METHOD
```

3. Description

Consider two random samples from two populations which have the same continuous distribution except for a shift in the location. Let the random sample, $x = (x_1,x_2,...,x_n)^T$, have distribution $F(x)$ and the random sample, $y = (y_1,y_2,...,y_m)^T$, have distribution $F(x-\theta)$.

G07EBF finds a point estimate, $\hat{\theta}$, of the difference in location θ together with an associated confidence interval. The estimates are based on the ordered differences $y_j - x_i$. The estimate $\hat{\theta}$ is defined by

$$\hat{\theta} = \text{median}\{y_j - x_i,\ i = 1,2,...,n; j = 1,2,...,m\}.$$

Let d_k for $k = 1,2,...,nm$ denote the nm (ascendingly) ordered differences $y_j - x_i$ for $i = 1,2,...,n; j = 1,2,...,m$. Then

$$\text{if } nm \text{ is odd, } \hat{\theta} = d_k \text{ where } k = \frac{nm-1}{2},$$

$$\text{if } nm \text{ is even, } \hat{\theta} = \frac{d_k + d_{k+1}}{2} \text{ where } k = \frac{nm}{2}.$$

This estimator arises from inverting the two sample Mann-Whitney rank test statistic, $U(\theta_0)$, for testing the hypothesis that $\theta = \theta_0$. Thus $U(\theta_0)$ is the value of the Mann-Whitney U statistic for the two independent samples $\{(x_i + \theta_0),\ i = 1,2...,n\}$ and $\{y_j,\ j = 1,2...,m\}$. Effectively $U(\theta_0)$ is a monotonically increasing step function of θ_0 with

$$\text{mean}(U) = \mu = \frac{nm}{2},$$

$$\text{var}(U) = \sigma^2 = \frac{nm(n+m+1)}{12}.$$

The estimate $\hat{\theta}$ is the solution to the equation $U(\hat{\theta}) = \mu$; two methods are available for solving this equation. These methods avoid the computation of all the ordered differences d_k; this is because for large n and m both the storage requirements and the computation time would be high.

The first is an exact method based on a set partitioning procedure on the set of all differences $y_j - x_i$ for $i = 1,2,...,n; j = 1,2,...,m$. This is adapted from the algorithm proposed by Monahan [3] for the computation of the Hodges-Lehmann estimator for a single population.

The second is an iterative algorithm, based on the Illinois method which is a modification of the regula falsi method, see McKean and Ryan [2]. This algorithm has proved suitable for the function $U(\theta_0)$ which is asymptotically linear as a function of θ_0.

The confidence interval limits are also based on the inversion of the Mann-Whitney test statistic.

Given a desired percentage for the confidence interval, $1-\alpha$, expressed as a proportion between 0.0 and 1.0 initial estimates of the upper and lower confidence limits for the Mann-Whitney U statistic are found;

$$U_l = \mu - 0.5 + (\sigma \times \Phi^{-1}(\alpha/2))$$
$$U_u = \mu + 0.5 + (\sigma \times \Phi^{-1}((1-\alpha)/2))$$

where Φ^{-1} is the inverse cumulative Normal distribution function.

U_l and U_u are rounded to the nearest integer values. These estimates are refined using an exact method, without taking ties into account, if $n+m \leq 40$ and $\max(n,m) \leq 30$ and a Normal approximation otherwise, to find U_l and U_u satisfying

$$P(U \leq U_l) \leq \alpha/2$$
$$P(U \leq U_l + 1) > \alpha/2$$

and

$$P(U \geq U_u) \leq \alpha/2$$
$$P(U \geq U_u - 1) > \alpha/2.$$

The function $U(\theta_0)$ is a monotonically increasing step function. It is the number of times a score in the second sample, y_j, precedes a score in the first sample, $x_i+\theta$, where we only count a half if a score in the second sample actually equals a score in the first.

Let $U_l = k$; then $\theta_l = d_{k+1}$. This is the largest value θ_l such that $U(\theta_l) = U_l$.

Let $U_u = nm - k$; then $\theta_u = d_{nm-k}$. This is the smallest value θ_u such that $U(\theta_u) = U_u$.

As in the case of $\hat{\theta}$, these equations may be solved using either the exact or iterative methods to find the values θ_l and θ_u.

Then (θ_l,θ_u) is the confidence interval for θ. The confidence interval is thus defined by those values of θ_0 such that the null hypothesis, $\theta = \theta_0$, is not rejected by the Mann-Whitney two sample rank test at the $(100\times\alpha)\%$ level.

4. References

[1] LEHMANN, E.L.
Nonparametrics. Statistical Methods Based on Ranks.
McGraw-Hill, New York, 1975.

[2] McKEAN, J.W. and RYAN, T.A.
Algorithm 516: An algorithm for obtaining confidence intervals and point estimates based on ranks in the two-sample location problem.
ACM Trans. Math. Softw., 3, pp. 183-185, 1977.

[3] MONAHAN, J.F.
Algorithm 616: Fast computation of the Hodges-Lehmann location estimator.
ACM Trans. Math. Softw., 10, pp. 265-270, 1984.

5. Parameters

1: METHOD – CHARACTER*1. *Input*

On entry: specifies the method to be used;

if METHOD = 'E' the exact algorithm is used,
if METHOD = 'A' the iterative algorithm is used.

Constraint: METHOD = 'E' or 'A'.

2: N – INTEGER. *Input*

On entry: the size of the first sample, n.

Constraint: N $\geq$ 1.

3: X(N) – ***real*** array. *Input*

On entry: the observations of the first sample, x_i for $i = 1,2,...,n$.

4: M – INTEGER. *Input*

On entry: the size of the second sample, m.

Constraint: M $\geq$ 1.

5: Y(M) – ***real*** array. *Input*

On entry: the observations of the second sample, y_j for $j = 1,2,...,m$.

6: CLEVEL – ***real***. *Input*

On entry: the confidence interval required, $1-\alpha$; e.g. for a 95% confidence interval set CLEVEL = 0.95.

Constraint: 0.0 < CLEVEL < 1.0.

7: THETA – ***real***. *Output*

On exit: the estimate of the difference in the location of the two populations, $\hat{\theta}$.

8: THETAL – ***real***. *Output*

On exit: the estimate of the lower limit of the confidence interval, θ_l.

9: THETAU – ***real***. *Output*

On exit: the estimate of the upper limit of the confidence interval, θ_u.

10: ESTCL – ***real***. *Output*

On exit: an estimate of the actual percentage confidence of the interval found, as a proportion between (0.0,1.0).

11: ULOWER – ***real***. *Output*

On exit: the value of the Mann-Whitney U statistic corresponding to the lower confidence limit, U_l.

12: UUPPER – ***real***. *Output*

On exit: the value of the Mann-Whitney U statistic corresponding to the upper confidence limit, U_u.

13: WRK(3*(M+N)) – ***real*** array. *Workspace*

14: IWRK(3*N) – INTEGER array. *Workspace*

15: IFAIL – INTEGER. *Input/Output*

On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

Errors detected by the routine:

If on entry IFAIL = 0 or –1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

IFAIL = 1

On entry, METHOD ≠ 'E' or 'A',
or N < 1,
or M < 1,
or CLEVEL ≤ 0.0,
or CLEVEL ≥ 1.0.

IFAIL = 2

Each sample consists of identical values. All estimates are set to the common difference between the samples.

IFAIL = 3

For at least one of the estimates $\hat{\theta}$, θ_l and θ_u, the underlying iterative algorithm (when METHOD = 'A') failed to converge. This is an unlikely exit but the estimate should still be a reasonable approximation.

7. Accuracy

The routine should return results accurate to 5 significant figures in the width of the confidence interval, that is the error for any one of the three estimates should be less than 0.00001×(THETAU−THETAL).

8. Further Comments

The time taken increases with the sample sizes n and m.

9. Example

The following program calculates a 95% confidence interval for the difference in location between the two populations from which the two samples of sizes 50 and 100 are drawn respectively.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G07EBF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX
      PARAMETER        (NMAX=100,MMAX=100)
*     .. Local Scalars ..
      real             CLEVEL, ESTCL, THETA, THETAL, THETAU, ULOWER,
     +                 UUPPER
      INTEGER          I, IFAIL, M, N
*     .. Local Arrays ..
      real             WRK(3*(NMAX+MMAX)), X(NMAX), Y(MMAX)
      INTEGER          IWRK(3*NMAX)
*     .. External Subroutines ..
      EXTERNAL         G07EBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G07EBF Example Program Results'
*     Skip Heading in data file
      READ (NIN,*)
      READ (NIN,*) N, M
      IF (N.LE.1 .OR. N.GT.NMAX .OR. M.LE.1 .OR. M.GT.MMAX) THEN
         WRITE (NOUT,99999) N, M
      ELSE
         READ (NIN,*)
         READ (NIN,*) (X(I),I=1,N)
         READ (NIN,*)
         READ (NIN,*) (Y(I),I=1,M)
         READ (NIN,*)
         READ (NIN,*) CLEVEL
         IFAIL = 0
*
         CALL G07EBF('Approx',N,X,M,Y,CLEVEL,THETA,THETAL,THETAU,ESTCL,
     +               ULOWER,UUPPER,WRK,IWRK,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Location estimator     Confidence Interval '
         WRITE (NOUT,*)
         WRITE (NOUT,99998) THETA, '( ', THETAL, ' , ', THETAU, ' )'
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Corresponding Mann-Whitney U statistics'
         WRITE (NOUT,*)
         WRITE (NOUT,99997) ' Lower : ', ULOWER
         WRITE (NOUT,99997) ' Upper : ', UUPPER
      END IF
      STOP
*
99999 FORMAT (4X,'N or M is out of range : N = ',I8,'  and M = ',I8)
99998 FORMAT (3X,F10.4,12X,A,F6.4,A,F6.4,A)
99997 FORMAT (A,F8.2)
      END
```

9.2. Program Data

```
G07EBF Example Program Data
 50 100
 First  sample  of N observations
 -0.582  0.157 -0.523 -0.769  2.338  1.664 -0.981  1.549  1.131 -0.460
 -0.484  1.932  0.306 -0.602 -0.979  0.132  0.256 -0.094  1.065 -1.084
 -0.969 -0.524  0.239  1.512 -0.782 -0.252 -1.163  1.376  1.674  0.831
  1.478 -1.486 -0.808 -0.429 -2.002  0.482 -1.584 -0.105  0.429  0.568
  0.944  2.558 -1.801  0.242  0.763 -0.461 -1.497 -1.353  0.301  1.941
 Second  sample  of M observations
  1.995  0.007  0.997  1.089  2.004  0.171  0.294  2.448  0.214  0.773
  2.960  0.025  0.638  0.937 -0.568 -0.711  0.931  2.601  1.121 -0.251
 -0.050  1.341  2.282  0.745  1.633  0.944  2.370  0.293  0.895  0.938
  0.199  0.812  1.253  0.590  1.522 -0.685  1.259  0.571  1.579  0.568
  0.381  0.829  0.277  0.656  2.497  1.779  1.922 -0.174  2.132  2.793
  0.102  1.569  1.267  0.490  0.077  1.366  0.056  0.605  0.628  1.650
  0.104  2.194  2.869 -0.171 -0.598  2.134  0.917  0.630  0.209  1.328
  0.368  0.756  2.645  1.161  0.347  0.920  1.256 -0.052  1.474  0.510
  1.386  3.550  1.392 -0.358  1.938  1.727 -0.372  0.911  0.499  0.066
  1.467  1.898  1.145  0.501  2.230  0.212  0.536  1.690  1.086  0.494
 Confidence Level
 0.95
```

9.3. Program Results

```
 G07EBF Example Program Results

  Location estimator     Confidence Interval

       0.9505            ( 0.5650 , 1.3050 )

  Corresponding Mann-Whitney U statistics

 Lower :  2007.00
 Upper :  2993.00
```