# NAG Library Function Document

# nag_mv_orthomax (g03bac)

## 1 Purpose

nag_mv_orthomax (g03bac) computes orthogonal rotations for a matrix of loadings using a generalized orthomax criterion.

## 2 Specification

```
#include <nag.h>
#include <nagg03.h>
void nag_mv_orthomax (Nag_RotationLoading stand, double g, Integer nvar,
     Integer k, double fl[], Integer tdf, double flr[], double r[],
     Integer tdr, double acc, Integer maxit, Integer *iter, NagError *fail)
```

## 3 Description

Let $\Lambda$ 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, $\lambda_{ij}^*$, of the rotated matrix, $\Lambda^*$, are either relatively large or small. The rotations may be found by minimizing the criterion:

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

where the constant $\gamma$ 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 standardized, 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 (1971), 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 (1971), but instead of the eigenvalue decomposition, the algorithm has been adapted to incorporate a singular value decomposition.

## 4 References

Cooley W C and Lohnes P R (1971) *Multivariate Data Analysis* Wiley

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

## 5 Arguments

1: **stand** – Nag_RotationLoading *Input*

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

**stand** = Nag_RoLoadStand
The loadings are row standardized.

**stand** = Nag_RoLoadNotStand
The loadings are left unstandardized.

*Constraint*: **stand** = Nag_RoLoadStand or Nag_RoLoadNotStand.

2: **g** – double *Input*

*On entry*: the criterion constant, $\gamma$, 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**[**nvar** $\times$ **tdf**] – double *Input/Output*

*On entry*: the matrix of loadings, $A$. **fl**$[(i-1) \times$ **tdf** $+ j - 1]$ must contain the loading for the $i$th variable on the $j$th factor, for $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, k$.

*On exit*: if **stand** = Nag_RoLoadStand the elements of **fl** are standardized 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** = Nag_RoLoadNotStand, **fl** will be unchanged on exit.

6: **tdf** – Integer *Input*

*On entry*: the stride separating matrix column elements in the arrays **fl**, **flr**.

*Constraint*: **tdf** $\geq$ **k**.

7: **flr**[**nvar** $\times$ **tdf**] – double *Output*

*On exit*: the rotated matrix of loadings, $A^*$. **flr**$[(i-1) \times$ **tdf** $+ j - 1]$ will contain the rotated loading for the $i$th variable on the $j$th factor, for $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, k$.

8: **r**[**k** $\times$ **tdr**] – double *Output*

**Note**: the $(i, j)$th element of the matrix $R$ is stored in **r**$[(i-1) \times$ **tdr** $+ j - 1]$.

*On exit*: the matrix of rotations, $R$.

9: **tdr** – Integer *Input*

*On entry*: the stride separating matrix column elements in the array **r**.

*Constraint*: **tdr** $\geq$ **k**.

10: **acc** – double *Input*

*On entry*: indicates the accuracy required. The iterative procedure of Cooley and Lohnes (1971) will be stopped and the final refinement computed when the change in $V$ is less than **acc** $\times \max(1.0, V)$. 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.

It is suggested that **acc** be set to 0.00001.

*Constraint*: **acc** $\geq 0.0$.

11: **maxit** – Integer *Input*

*On entry*: the maximum number of iterations. It is suggested that **maxit** be set to 30.

*Constraint*: **maxit** $\geq 1$.

12: **iter** – Integer * *Output*

*On exit*: the number of iterations performed.

13: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

**NE_2_INT_ARG_LT**

On entry, **nvar** $= \langle value \rangle$ while **k** $= \langle value \rangle$. These arguments must satisfy **nvar** $\geq$ **k**.

On entry, **tdf** $= \langle value \rangle$ while **k** $= \langle value \rangle$. These arguments must satisfy **tdf** $\geq$ **k**.

On entry, **tdr** $= \langle value \rangle$ while **k** $= \langle value \rangle$. These arguments must satisfy **tdr** $\geq$ **k**.

**NE_ACC_ITER**

The algorithm to find R has failed to reach the required accuracy in the given number of iterations, $\langle value \rangle$. Try increasing **acc** or increasing **maxit**. The returned solution should be a reasonable approximation.

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, argument **stand** had an illegal value.

**NE_INT_ARG_LE**

On entry, **maxit** $= \langle value \rangle$.
Constraint: **maxit** $> 0$.

**NE_INT_ARG_LT**

On entry, **k** $= \langle value \rangle$.
Constraint: **k** $\geq 2$.

**NE_INTERNAL_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

**NE_REAL_ARG_LT**

On entry, **acc** must not be less than 0.0: **acc** $= \langle value \rangle$.

On entry, **g** must not be less than 0.0: **g** $= \langle value \rangle$.

**NE_SVD_NOT_CONV**

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

## 7 Accuracy

The accuracy is determined by the value of **acc**.

## 8 Parallelism and Performance

nag_mv_orthomax (g03bac) is not threaded in any implementation.

## 9 Further Comments

If the results of a principal component analysis as carried out by nag_mv_prin_comp (g03aac) are to be rotated, the loadings as returned in the array $p$ by nag_mv_prin_comp (g03aac) can be supplied via the argument **fl** to nag_mv_orthomax (g03bac). The resulting rotation matrix can then be used to rotate the principal component scores as returned in the array $v$ by nag_mv_prin_comp (g03aac). The function nag_dgemm (f16yac) may be used for this matrix multiplication.

## 10 Example

The example is taken from page 75 of Lawley and Maxwell (1971). The results from a factor analysis of ten variables using three factors are input and rotated using varimax rotations without standardizing rows.

### 10.1 Program Text

```
/* nag_mv_orthomax (g03bac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg03.h>

#define R(I, J)   r[(I) *tdr + J]
#define FL(I, J)  fl[(I) *tdfl + J]
#define FLR(I, J) flr[(I) *tdfl + J]

int main(void)
{
  Integer exit_status = 0, i, iter, j, k, maxit, nvar, tdfl, tdr;
  double acc, *fl = 0, *flr = 0, g, *r = 0;
  char nag_enum_arg[40];
  Nag_RotationLoading stand;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_mv_orthomax (g03bac) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &nvar);
#else
```

```
  scanf("%" NAG_IFMT "", &nvar);
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &k);
#else
  scanf("%" NAG_IFMT "", &k);
#endif
#ifdef _WIN32
  scanf_s("%lf", &g);
#else
  scanf("%lf", &g);
#endif
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
#ifdef _WIN32
  scanf_s("%lf", &acc);
#else
  scanf("%lf", &acc);
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &maxit);
#else
  scanf("%" NAG_IFMT "", &maxit);
#endif

  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  stand = (Nag_RotationLoading) nag_enum_name_to_value(nag_enum_arg);

  if (k >= 2 && nvar >= k) {
    if (!(r = NAG_ALLOC(k * k, double)) ||
        !(fl = NAG_ALLOC(nvar * k, double)) ||
        !(flr = NAG_ALLOC(nvar * k, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    tdr = k;
    tdfl = k;
  }
  else {
    printf("Invalid k or nvar.\n");
    exit_status = 1;
    return exit_status;
  }
  for (i = 0; i < nvar; ++i) {
    for (j = 0; j < k; ++j)
#ifdef _WIN32
      scanf_s("%lf", &FL(i, j));
#else
      scanf("%lf", &FL(i, j));
#endif
  }
  /* nag_mv_orthomax (g03bac).
   * Orthogonal rotations for loading matrix
   */
  nag_mv_orthomax(stand, g, nvar, k, fl, tdfl, flr, r, tdr, acc, maxit, &iter,
                  &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_mv_orthomax (g03bac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\n     Rotated factor loadings\n\n");
  for (i = 0; i < nvar; ++i) {
```

```
    for (j = 0; j < k; ++j)
      printf("  %8.3f", FLR(i, j));
    printf("\n");
  }
  printf("\n     Rotation matrix\n\n");
  for (i = 0; i < k; ++i) {
    for (j = 0; j < k; ++j)
      printf("  %8.3f", R(i, j));
    printf("\n");
  }

END:
  NAG_FREE(r);
  NAG_FREE(fl);
  NAG_FREE(flr);

  return exit_status;
}
```

## 10.2 Program Data

```
nag_mv_orthomax (g03bac) Example Program Data
  10 3 1.0 Nag_RoLoadNotStand 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
```

## 10.3 Program Results

```
nag_mv_orthomax (g03bac) 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
```