NAG C Library Function Document

nag_moments_ratio_quad_forms (g01nbc)

1 Purpose

nag_moments_ratio_quad_forms (g01nbc) computes the moments of ratios of quadratic forms in Normal variables and related statistics.

2 Specification

```c
void nag_moments_ratio_quad_forms (Nag_OrderType order, Nag_MomentType ratio_type,
                               Nag_IncludeMean mean, Integer n, const double a[], Integer pda,
                               const double b[], Integer pdb, const double c[], Integer pdc,
                               const double ela[], const double emu[], const double sigma[],
                               Integer pdsig, Integer l1, Integer l2, Integer *lmax,
                               double rmom[], double *abserr, double eps, NagError *fail)
```

3 Description

Let \( x \) have an \( n \)-dimensional multivariate Normal distribution with mean \( \mu \) and variance-covariance matrix \( \Sigma \). Then for a symmetric matrix \( A \) and symmetric positive semi-definite matrix \( B \), nag_moments_ratio_quad_forms (g01nbc) computes a subset, \( l_1 \) to \( l_2 \), of the first 12 moments of the ratio of quadratic forms

\[ R = x^T Ax / x^T B x. \]

The \( s \)th moment (about the origin) is defined as

\[ E(R^s), \]

where \( E \) denotes the expectation. Alternatively, this function will compute the following expectations:

\[ E(R^s(a^T x)) \] (2)

and

\[ E(R^s(x^T C x)), \] (3)

where \( a \) is a vector of length \( n \) and \( C \) is a \( n \) by \( n \) symmetric matrix, if they exist. In the case of (2) the moments are zero if \( \mu = 0 \).

The conditions of theorems 1, 2 and 3 of Magnus (1986) and Magnus (1990) are used to check for the existence of the moments. If all the requested moments do not exist, the computations are carried out for those moments that are requested up to the maximum that exist, \( l_{MAX} \).

This function is based on the routine QRMOM written by Magnus and Pesaran (1993) and based on the theory given by Magnus (1986) and Magnus (1990). The computation of the moments requires first the computation of the eigenvectors of the matrix \( L^T B L \), where \( LL^T = \Sigma \). The matrix \( L^T B L \) must be positive semi-definite and not null. Given the eigenvectors of this matrix, a function which has to be integrated over the range zero to infinity can be computed. This integration is performed using nag_1d_quad_inf (d01amc).

4 References


5 Parameters

1: \texttt{order} – \texttt{Nag\_OrderType} \hspace{2cm} \textit{Input}

\textit{On entry:} the \texttt{order} parameter specifies the two-dimensional storage scheme being used, i.e., row-
major ordering or column-major ordering. C language defined storage is specified by \texttt{order = Nag\_RowMajor}. See Section 2.2.1.4 of the Essential Introduction for a more detailed
explanation of the use of this parameter.

\textit{Constraint:} \texttt{order = Nag\_RowMajor} or \texttt{Nag\_ColMajor}.

2: \texttt{ratio\_type} – \texttt{Nag\_MomentType} \hspace{2cm} \textit{Input}

\textit{On entry:} indicates the moments of which function are to be computed.

If \texttt{ratio\_type = Nag\_RatioMoments} (Ratio), \(E(R^s)\) is computed.

If \texttt{ratio\_type = Nag\_LinearRatio} (Linear with ratio), \(E(R^s(a^T x))\) is computed.

If \texttt{ratio\_type = Nag\_QuadRatio} (Quadratic with ratio), \(E(R^s(x^T C x))\) is computed.

\textit{Constraint:} \texttt{ratio\_type = Nag\_RatioMoments, Nag\_LinearRatio} or \texttt{Nag\_QuadRatio}.

3: \texttt{mean} – \texttt{Nag\_IncludeMean} \hspace{2cm} \textit{Input}

\textit{On entry:} indicates if the mean, \(\mu\), is zero.

If \texttt{mean = Nag\_MeanZero}, \(\mu\) is zero.

If \texttt{mean = Nag\_MeanInclude}, the value of \(\mu\) is supplied in \texttt{emu}.

\textit{Constraint:} \texttt{mean = Nag\_MeanZero} or \texttt{Nag\_MeanInclude}.

4: \texttt{n} – Integer \hspace{2cm} \textit{Input}

\textit{On entry:} the dimension of the quadratic form, \(n\).

\textit{Constraint:} \texttt{n > 1}.

5: \texttt{a[dim]} – const double \hspace{2cm} \textit{Input}

\textit{Note:} the dimension, \texttt{dim}, of the array \texttt{a} must be at least \texttt{pda} \times \texttt{n}.

If \texttt{order = Nag\_ColMajor}, the \((i, j)\)th element of the matrix \(A\) is stored in \texttt{a}[(\texttt{j} - 1) \times \texttt{pda} + i - 1] and if \texttt{order = Nag\_RowMajor}, the \((i, j)\)th element of the matrix \(A\) is stored in \texttt{a}[(\texttt{i} - 1) \times \texttt{pda} + j - 1].

\textit{On entry:} the \(n\) by \(n\) symmetric matrix \(A\). Only the lower triangle is referenced.

6: \texttt{pda} – Integer \hspace{2cm} \textit{Input}

\textit{On entry:} the stride separating matrix row or column elements (depending on the value of \texttt{order}) in the
array \texttt{a}.

\textit{Constraint:} \texttt{pda} \geq \texttt{n}.

7: \texttt{b[dim]} – const double \hspace{2cm} \textit{Input}

\textit{Note:} the dimension, \texttt{dim}, of the array \texttt{b} must be at least \texttt{pdb} \times \texttt{n}.

If \texttt{order = Nag\_ColMajor}, the \((i, j)\)th element of the matrix \(B\) is stored in \texttt{b}[(\texttt{j} - 1) \times \texttt{pdb} + i - 1] and if \texttt{order = Nag\_RowMajor}, the \((i, j)\)th element of the matrix \(B\) is stored in \texttt{b}[(\texttt{i} - 1) \times \texttt{pdb} + j - 1].
On entry: the \( n \) by \( n \) positive semi-definite symmetric matrix \( B \). Only the lower triangle is referenced.

Constraint: the matrix \( B \) must be positive semi-definite.

8: \( \text{pdb} \) – Integer

On entry: the stride separating matrix row or column elements (depending on the value of \texttt{order}) in the array \( b \).

Constraint: \( \text{pdb} \geq n \).

9: \( c[\text{dim}] \) – const double

Note: the dimension, \( \text{dim} \), of the array \( c \) must be at least
\[
\max(1, \text{pdc} \times n) \quad \text{when} \quad \text{ratio_type} = \text{Nag QuadRatio};
\]
\[
1 \quad \text{otherwise}.
\]

If \( \text{order} = \text{Nag ColMajor} \), the \((i, j)\)th element of the matrix \( C \) is stored in \( c[(j - 1) \times \text{pdc} + i - 1] \) and
if \( \text{order} = \text{Nag RowMajor} \), the \((i, j)\)th element of the matrix \( C \) is stored in \( c[(i - 1) \times \text{pdc} + j - 1] \).

On entry: if \( \text{ratio_type} = \text{Nag QuadRatio} \), \( c \) must contain the \( n \) by \( n \) symmetric matrix \( C \); only the lower triangle is referenced. If \( \text{ratio_type} \neq \text{Nag QuadRatio} \), \( c \) is not referenced.

10: \( \text{pdc} \) – Integer

On entry: the stride separating matrix row or column elements (depending on the value of \texttt{order}) in the array \( c \).

Constraints:
\[
\text{if} \quad \text{ratio_type} = \text{Nag QuadRatio}, \quad \text{pdc} \geq n; \\
\text{otherwise} \quad \text{pdc} \geq 1.
\]

11: \( \text{ela}[\text{dim}] \) – const double

Note: the dimension, \( \text{dim} \), of the array \( \text{ela} \) must be at least \( n \) when \( \text{ratio_type} = \text{Nag LinearRatio} \)
and at least 1 otherwise.

On entry: if \( \text{ratio_type} = \text{Nag LinearRatio} \), \( \text{ela} \) must contain the vector \( a \) of length \( n \), otherwise \( a \) is not referenced.

12: \( \text{emu}[\text{dim}] \) – const double

Note: the dimension, \( \text{dim} \), of the array \( \text{emu} \) must be at least \( n \) when \( \text{mean} = \text{Nag MeanInclude} \)
and at least 1 otherwise.

On entry: if \( \text{mean} = \text{Nag MeanInclude} \), \( \text{emu} \) must contain the \( n \) elements of the vector \( \mu \). If \( \text{mean} = \text{Nag MeanZero} \), \( \text{emu} \) is not referenced.

13: \( \text{sigma}[\text{dim}] \) – const double

Note: the dimension, \( \text{dim} \), of the array \( \text{sigma} \) must be at least \( \text{pdsig} \times n \).

If \( \text{order} = \text{Nag ColMajor} \), the \((i, j)\)th element of the matrix is stored in \( \text{sigma}[(j - 1) \times \text{pdsig} + i - 1] \)
and if \( \text{order} = \text{Nag RowMajor} \), the \((i, j)\)th element of the matrix is stored in \( \text{sigma}[(i - 1) \times \text{pdsig} + j - 1] \).

On entry: the \( n \) by \( n \) variance-covariance matrix \( \Sigma \). Only the lower triangle is referenced.

Constraint: the matrix \( \Sigma \) must be positive-definite.

14: \( \text{pdsig} \) – Integer

On entry: the stride separating matrix row or column elements (depending on the value of \texttt{order}) in the array \( \text{sigma} \).

Constraint: \( \text{pdsig} \geq n \).
15: \( l_1 \) – Integer

\textit{Input}

\textit{On entry}: the first moment to be computed, \( l_1 \).

\textit{Constraint}: \( 0 < l_1 \leq l_2 \).

16: \( l_2 \) – Integer

\textit{Input}

\textit{On entry}: the last moment to be computed, \( l_2 \).

\textit{Constraint}: \( l_1 \leq l_2 \leq 12 \).

17: \( l_{\text{max}} \) – Integer *

\textit{Output}

\textit{On exit}: the highest moment computed, \( l_{\text{MAX}} \). This will be \( l_2 \) on successful exit.

18: \( r\text{mom}[l_2 - l_1 + 1] \) – double

\textit{Output}

\textit{On exit}: the \( l_1 \) to \( l_{\text{MAX}} \) moments.

19: \( \text{abserr} \) – double *

\textit{Output}

\textit{On exit}: the estimated maximum absolute error in any computed moment.

20: \( \text{eps} \) – double

\textit{Input}

\textit{On entry}: the relative accuracy required for the moments, this value is also used in the checks for the existence of the moments. If \( \text{eps} = 0.0 \), a value of \( \sqrt{\epsilon} \) where \( \epsilon \) is the \textit{machine precision} used.

\textit{Constraint}: \( \text{eps} = 0.0 \) or \( \text{eps} \geq \text{machine precision} \).

21: \( \text{fail} \) – NagError *

\textit{Input}/\textit{Output}

The NAG error parameter (see the Essential Introduction).

6 \textbf{Error Indicators and Warnings}

\textbf{NE_INT}

\textit{On entry}, \( n = \langle \text{value} \rangle \).

\textit{Constraint}: \( n > 1 \).

\textit{On entry}, \( \text{pda} = \langle \text{value} \rangle \).

\textit{Constraint}: \( \text{pda} > 0 \).

\textit{On entry}, \( \text{pdb} = \langle \text{value} \rangle \).

\textit{Constraint}: \( \text{pdb} > 0 \).

\textit{On entry}, \( \text{pdc} = \langle \text{value} \rangle \).

\textit{Constraint}: \( \text{pdc} > 0 \).

\textit{On entry}, \( \text{pdsig} = \langle \text{value} \rangle \).

\textit{Constraint}: \( \text{pdsig} > 0 \).

\textit{On entry}, \( l_2 = \langle \text{value} \rangle \).

\textit{Constraint}: \( l_2 \leq 12 \).

\textit{On entry}, \( l_1 = \langle \text{value} \rangle \).

\textit{Constraint}: \( l_1 \geq 1 \).

\textbf{NE_INT_2}

\textit{On entry}, \( \text{pda} = \langle \text{value} \rangle, \ n = \langle \text{value} \rangle \).

\textit{Constraint}: \( \text{pda} \geq n \).

\textit{On entry}, \( \text{pdb} = \langle \text{value} \rangle, \ n = \langle \text{value} \rangle \).

\textit{Constraint}: \( \text{pdb} \geq n \).
On entry, \( pdsig = \langle\text{value}\rangle, \ n = \langle\text{value}\rangle \).
Constraint: \( pdsig \geq n \).

On entry, \( l2 < l1; l1 = \langle\text{value}\rangle, l2 = \langle\text{value}\rangle \).
On entry, \( pdc < n; pdc = \langle\text{value}\rangle, n = \langle\text{value}\rangle \).

\textbf{NE_ENUM_INT_2}

On entry, \( \text{ratio}\_\text{type} = \langle\text{value}\rangle, \ n = \langle\text{value}\rangle, pdc = \langle\text{value}\rangle \).
Constraint: if \( \text{ratio}\_\text{type} = \text{Nag}_\text{QuadRatio}, \ pdc \geq n \); otherwise \( pdc \geq 1 \).

\textbf{NE_ACCURACY}

Full accuracy not achieved in integration.

\textbf{NE_EIGENVALUES}

Failure in computing eigenvalues.

\textbf{NE_MOMENTS}

Only \( \langle\text{value}\rangle \) moments exist, less than \( l1 = \langle\text{value}\rangle \).

\textbf{NE_POS_DEF}

On entry, \( \text{sigma} \) is not positive-definite.

\textbf{NE_POS_SEMI_DEF}

The matrix \( LBL \) is not positive semi-definite or is null.
On entry, \( \text{b} \) is not positive semi-definite or is null.

\textbf{NE_REAL}

On entry, if \( \text{eps} \neq 0, \ \text{eps} < \text{machine precision}; \ \text{eps} = \langle\text{value}\rangle \).

\textbf{NE_SOME_MOMENTS}

Only \( \langle\text{value}\rangle \) moments exist, less than \( l2 = \langle\text{value}\rangle \).

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

On entry, parameter \( \langle\text{value}\rangle \) had an illegal value.

\textbf{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 consult NAG for assistance.

7 Accuracy

The relative accuracy is specified by \( \text{eps} \) and an estimate of the maximum absolute error for all computed moments is returned in \( \text{abserr} \).

8 Further Comments

None.
9 Example

The example is given by Magnus and Pesaran (1993) and considers the simple autoregression:

\[ y_t = \beta y_{t-1} + u_t, \quad t = 1, 2, \ldots, n, \]

where \( \{u_t\} \) is a sequence of independent Normal variables with mean zero and variance one, and \( y_0 \) is known. The least-squares estimate of \( \beta \), \( \hat{\beta} \), is given by

\[ \hat{\beta} = \frac{\sum_{t=2}^{n} y_t y_{t-1}}{\sum_{t=2}^{n} y_t^2}. \]

Thus \( \hat{\beta} \) can be written as a ratio of quadratic forms and its moments computed using nag_moments_ratio_quad_forms (g01nbc). The matrix \( A \) is given by

\[ A(i+1,i) = \frac{1}{2}, \quad i = 1, 2, \ldots n - 1; \]
\[ A(i,j) = 0, \quad \text{otherwise}, \]

and the matrix \( B \) is given by

\[ B(i,i) = 1, \quad i = 1, 2, \ldots n - 1; \]
\[ B(i,j) = 0, \quad \text{otherwise}. \]

The value of \( \Sigma \) can be computed using the relationships

\[ \text{var}(y_t) = \beta^2 \text{var}(y_{t-1}) + 1 \]

and

\[ \text{cov}(y_t y_{t+k}) = \beta \text{cov}(y_t y_{t+k-1}) \]

for \( k \geq 0 \) and \( \text{var}(y_1) = 1. \)

The values of \( \beta, y_0, n, \) and the number of moments required are read in and the moments computed and printed.

9.1 Program Text

/* nag_moments_ratio_quad_forms (g01nbc) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
*
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nag01.h>

int main(void)
{
    /* Scalars */
    double abserr, beta, y0, eps;
    Integer exit_status, i, j, l1, l2, lmax, n, pda, pdb, pdsigma;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    double *a=0, *b=0, *c=0, *ela=0, *emu=0, *rmom=0, *sigma=0;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda + I - 1]
    #define B(I,J) b[(J-1)*pdb + I - 1]
    #define SIGMA(I,J) sigma[(J-1)*pdsigma + I - 1]
    #define order = Nag_ColMajor;
    #endif

    ...
```c
#define A(I,J) a[(I-1)*pda + J - 1]
#define B(I,J) b[(I-1)*pdb + J - 1]
#define SIGMA(I,J) sigma[(I-1)*pdsigma + J - 1]

order = Nag_RowMajor;
#endif

INIT_FAIL(fail);
exit_status = 0;
Vprintf("g01nbc Example Program Results\n");

/* Skip heading in data file */
Vscanf("%*[\n"]
);
Vscanf("%lf%lf%*[\n"] , &beta, &y0);
Vscanf("%ld%ld%ld%*[\n"] , &n, &l1, &l2);

/* Allocate memory */
if ( !(a = NAG_ALLOC(n * n, double)) |
    !(b = NAG_ALLOC(n * n, double)) |
    !(c = NAG_ALLOC(n * n, double)) |
    !(ela = NAG_ALLOC(n, double)) |
    !(emu = NAG_ALLOC(n, double)) |
    !(rmom = NAG_ALLOC(l2-l1+1, double)) |
    !(sigma = NAG_ALLOC(n* n, double)) ) |
{ Vprintf("Allocation failure\n");
  exit_status = -1;
  goto END;
}
pda = n;
pdb = n;
pdsigma = n;

/* Compute A, EMU, and SIGMA for simple autoregression */
for (i = 1; i <= n; ++i)
{ for (j = i; j <= n; ++j)
  { A(j, i) = 0.0;
    B(j, i) = 0.0;
  }
}
for (i = 1; i <= n - 1; ++i)
{ A(i + 1, i) = 0.5;
  B(i, i) = 1.0;
}
emu[0] = y0 * beta;
for (i = 1; i <= n - 1; ++i)
  emu[i] = beta * emu[i - 1];
SIGMA(1, 1) = 1.0;
for (i = 2; i <= n; ++i)
  SIGMA(i, i) = beta * beta * SIGMA(i - 1, i - 1) + 1.0;
for (i = 1; i <= n; ++i)
{ for (j = i + 1; j <= n; ++j)
    SIGMA(j, i) = beta * SIGMA(j - 1, i);
}
eps = 0.0;
g01nbc(order, Nag_RatioMoments, Nag_MeanInclude, n,
a, n, b, n, c, n, ela, emu, sigma, n, l1, l2,
&lmax, rmom, &abserr, eps, &fail);
if (fail.code == NE_NOERROR || fail.code == NE_SOME_MOMENTS
  || fail.code == NE_ACCURACY)
{ Vprintf("\n");
  Vprintf(" n = %3ld beta = %6.3f y0 = %6.3f\n", n, beta, y0);
```
Vprintf("\n");
Vprintf(" Moments\n");
Vprintf("\n");

j = 0;
for (i = l1; i <= lmax; ++i)
{
    ++j;
    Vprintf("%3ld%12.3e\n", i, rmom[j - 1]);
}

else
{
    Vprintf("Error from g01nbc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
if (c) NAG_FREE(c);
if (ela) NAG_FREE(ela);
if (emu) NAG_FREE(emu);
if (rmom) NAG_FREE(rmom);
if (sigma) NAG_FREE(sigma);

    return exit_status;
}

9.2 Program Data

    g01nbc Example Program Data

    0.8 1.0 : Beta Y0

    10 1 3 : N L1 L1

9.3 Program Results

    g01nbc Example Program Results

    n = 10 beta = 0.800 y0 = 1.000

    Moments

    1 6.820e-01
    2 5.357e-01
    3 4.427e-01