

NAG C Library Function Document

nag_multi_normal (g01hbc)

1 Purpose

nag_multi_normal (g01hbc) returns the upper tail, lower tail or central probability associated with a multivariate Normal distribution of up to ten dimensions.

2 Specification

```
#include <nag.h>
#include <nagg01.h>

double nag_multi_normal (Nag_TailProbability tail, Integer n,
    const double a[], const double b[], const double mean[],
    const double sigma[], Integer tdsig, double tol, Integer maxpts,
    NagError *fail)
```

3 Description

Let the vector random variable $X = (X_1, X_2, \dots, X_n)^T$ follow a n dimensional multivariate Normal distribution with mean vector μ and n by n variance-covariance matrix Σ , then the probability density function, $f(X : \mu, \Sigma)$, is given by

$$f(X : \mu, \Sigma) = (2\pi)^{-(1/2)n} |\Sigma|^{-1/2} \exp\{-\frac{1}{2}(X - \mu)^T \Sigma^{-1} (X - \mu)\}.$$

The lower tail probability is defined by:

$$P(X_1 \leq b_1, \dots, X_n \leq b_n : \mu, \Sigma) = \int_{-\infty}^{b_1} \dots \int_{-\infty}^{b_n} f(X : \mu, \Sigma) dX_n \dots dX_1.$$

The upper tail probability is defined by:

$$P(X_1 \geq a_1, \dots, X_n \geq a_n : \mu, \Sigma) = \int_{a_1}^{\infty} \dots \int_{a_n}^{\infty} f(X : \mu, \Sigma) dX_n \dots dX_1.$$

The central probability is defined by:

$$P(a_1 \leq X_1 \leq b_1, \dots, a_n \leq X_n \leq b_n : \mu, \Sigma) = \int_{a_1}^{b_1} \dots \int_{a_n}^{b_n} f(X : \mu, \Sigma) dX_n \dots dX_1.$$

To evaluate the probability for $n \geq 3$, the probability density function of $X_1, X_2, \dots, X_n$ is considered as the product of the conditional probability of $X_1, X_2, \dots, X_{n-2}$ given X_{n-1} and X_n and the marginal bivariate Normal distribution of X_{n-1} and X_n . The bivariate Normal probability can be evaluated as described in nag_bivariate_normal_dist (g01hac) and numerical integration is then used over the remaining $n - 2$ dimensions. In the case of $n = 3$ the routine nag_1d_quad_gen_1 (d01sjc) is used and for $n > 3$ the routine nag_multid_quad_adapt_1 (d01wcc) is used.

To evaluate the probability for $n = 1$ a direct call to nag_prob_normal (g01eac) is made and for $n = 2$ calls to nag_bivariate_normal_dist (g01hac) are made.

4 Parameters

- 1: **tail** – Nag_TailProbability *Input*
On entry: indicates which probability is to be returned.
 If **tail** = Nag_LowerTail, the lower tail probability is returned.
 If **tail** = Nag_UpperTail, the upper tail probability is returned.
 If **tail** = Nag_Central, the central probability is returned.
Constraint: **tail** = Nag_Central, Nag_LowerTail or Nag_UpperTail.

- 2: **n** – Integer *Input*
On entry: the number of dimensions, n .
Constraint: $1 \leq \mathbf{n} \leq 10$.

- 3: **a[n]** – const double *Input*
On entry: if **tail** = Nag_Central or Nag_UpperTail, the lower bounds, a_i , for $i = 1, 2, \dots, n$.
 If **tail** = Nag_LowerTail, **a** is not referenced.

- 4: **b[n]** – const double *Input*
On entry: if **tail** = Nag_Central or Nag_LowerTail, the upper bounds, b_i , for $i = 1, 2, \dots, n$.
 If **tail** = Nag_UpperTail, **b** is not referenced.
Constraint: if **tail** = Nag_Central, $\mathbf{a}(i) < \mathbf{b}(i)$, for $i = 1, 2, \dots, n$.

- 5: **mean[n]** – const double *Input*
On entry: the mean vector, μ , of the multivariate Normal distribution.

- 6: **sigma[n][tdsig]** – const double *Input*
On entry: the variance-covariance matrix, Σ , of the multivariate Normal distribution. Only the lower triangle is referenced.
Constraint: Σ must be positive-definite.

- 7: **tdsig** – Integer *Input*
On entry: the second dimension of the array **sigma** as declared in the function from which nag_multi_normal is called.
Constraint: **tdsig** $\geq \mathbf{n}$.

- 8: **tol** – double *Input*
On entry: if $n > 2$, the relative accuracy required for the probability, and if the upper or the lower tail probability is requested, then **tol** is also used to determine the cut-off points (see Section 6.1).
 If $n = 1$, **tol** is not referenced.
Suggested value: **tol** = 0.0001.
Constraint: if $\mathbf{n} > 1$, **tol** > 0.0 .

- 9: **maxpts** – Integer *Input*
On entry: the maximum number of sub-intervals or integrand evaluations.
 If $n = 3$, then the maximum number of sub-intervals used by nag_ld_quad_gen_1 (d01sjc) is **maxpts**/4. Note however increasing **maxpts** above 1000 will not increase the maximum number of sub-intervals above 250.

If $n > 3$ the maximum number of integrand evaluations used by nag_multid_quad_adapt_1 (d01wcc) is $\alpha(\mathbf{maxpts}/n - 1)$, where $\alpha = 2^{n-2} + 2(n-2)^2 + 2(n-2) + 1$.

If $n = 1$ or 2 , then **maxpts** will not be used.

Suggested value: 2000 if $n > 3$ and 1000 if $n = 3$.

Constraints:

maxpts ≥ 1 , for **n** = 1 or 2,

maxpts $\geq 4 \times \mathbf{n}$, for **n** ≥ 3 .

10: **fail** – NagError *

Input/Output

The NAG error parameter (see the Essential Introduction).

5 Error Indicators and Warnings

NE_INT_ARG_CONS

On entry, **n** = *<value>*.

This parameter must satisfy $1 \leq \mathbf{n} \leq 10$.

On entry, **maxpts** = *<value>*.

This parameter must satisfy **maxpts** $\geq 4 \times \mathbf{n}$, if **n** ≥ 3 .

NE_2_INT_ARG_LT

On entry, **tdsig** = *<value>* while **n** = *<value>*.

These parameters must satisfy **tdsig** $\geq \mathbf{n}$.

NE_REAL_ARG_CONS

On entry, **tol** = *<value>*.

This parameter must satisfy **tol** > 0 , if **n** > 1 .

NE_2_REAL_ARRAYS_CONS

On entry, **a**[*<value>*] = *<value>* while **b**[*<value>*] = *<value>*.

Constraint: if **tail** = **Nag_Central**, **a**[*i*] $< \mathbf{b}[i]$, for $i = 0, 1, \dots, n-1$.

NE_BAD_PARAM

On entry, parameter **tail** had an illegal value.

NE_ALLOC_FAIL

Memory allocation failed.

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.

NE_POS_DEF

The matrix **sigma** is not positive definite.

NE_ACC

The requested accuracy cannot be achieved. Try increasing **tol**. The returned result is an approximation to the required result.

NE_ROUND_OFF

Round-off error prevents the requested accuracy from being achieved. Try increasing **tol**. The returned result is an approximation to the required result. This result will only occur if **n** = 3.

6 Further Comments

The time taken is related to both the number of dimensions, the range over which the integration takes place ($b_i - a_i$, for $i = 1, 2, \dots, n$) and the value of Σ as well as the accuracy required. As the numerical integration does not take place over the last two dimensions speed may be improved by arranging X so that the largest ranges of integration are for X_{n-1} and X_n .

6.1 Accuracy

The accuracy should be as specified by **tol**. For the upper and lower tail probabilities the infinite limits are approximated by cut-off points for the $n - 2$ dimensions over which the numerical integration takes place; these cut-off points are given by $\Phi^{-1}(\text{tol}/(10 \times n))$, where Φ^{-1} is the inverse univariate Normal distribution function.

6.2 References

Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* Griffin (3rd Edition)

7 See Also

nag_ld_quad_gen_1 (d01sjc)
 nag_multid_quad_adapt_1 (d01wcc)
 nag_prob_normal (g01eac)
 nag_bivariate_normal_dist (g01hac)

8 Example

The mean and variance of a four-dimensional multivariate Normal distribution are input and a central probability computed and printed.

8.1 Program Text

```
/* nag_multi_normal (g01hbc) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
    char tail[2];
    double *a=0, *b=0, prob, *sigma=0, tol, *mean=0;
    Integer i, j, maxpts, n;
    Integer exit_status=0;
    NagError fail;
    Nag_TailProbability tail_enum;

#define SIGMA(I,J) sigma[((I)-1)*n + (J)-1]
```

```

INIT_FAIL(fail);
Vprintf("g01hbc Example Program Results\n");

/* Skip heading in data file */
Vscanf("%*[^\\n]");

Vscanf("%ld %lf %s", &n, &tol, tail);
if (!(a = NAG_ALLOC(n, double))
    || !(b = NAG_ALLOC(n, double))
    || !(mean = NAG_ALLOC(n, double))
    || !(sigma = NAG_ALLOC(n*n, double)))
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

if (*tail == 'L')
    tail_enum = Nag_LowerTail;
else if (*tail == 'U')
    tail_enum = Nag_UpperTail;
else if (*tail == 'C')
    tail_enum = Nag_Central;
else
    tail_enum = (Nag_TailProbability)-999;

for (j = 1; j <= n; ++j)
    Vscanf("%lf", &mean[j - 1]);
for (i = 1; i <= n; ++i)
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &SIGMA(i,j));
if (tail_enum == Nag_Central || tail_enum == Nag_UpperTail)
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &a[j - 1]);
if (tail_enum == Nag_Central || tail_enum == Nag_LowerTail)
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &b[j - 1]);
maxpts = 2000;
prob = g01hbc(tail_enum, n, a, b, mean, sigma, n, tol, maxpts, &fail);
if (fail.code == NE_NOERROR || fail.code == NE_POS_DEF
    || fail.code == NE_ACC || fail.code == NE_ROUND_OFF )
{
    Vprintf("%s %6.4f\\n", "\\nMultivariate Normal probability = ", prob);
}
else
{
    Vprintf("Error from g01hbc.\\n%s\\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
if (mean) NAG_FREE(mean);
if (sigma) NAG_FREE(sigma);
return exit_status;
}

```

8.2 Program Data

g01hbc Example Program Data

4 0.0001 C

0.0 0.0 0.0 0.0

1.0 0.9 0.9 0.9

0.9 1.0 0.9 0.9

0.9 0.9 1.0 0.9

0.9 0.9 0.9 1.0

-2.0 -2.0 -2.0 -2.0

2.0 2.0 2.0 2.0

8.3 Program Results

g01hbc Example Program Results

Multivariate Normal probability = 0.9142
