NAG 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 $X = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$ follow an 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

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

The lower tail probability is defined by:

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

The upper tail probability is defined by:

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

The central probability is defined by:

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

To evaluate the probability for $n \geq 3$, the probability density function of $X_1, X_2, \ldots, X_n$ is considered as the product of the conditional probability of $X_1, X_2, \ldots, X_{n-2}$ 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 $n = 3$, nag_1d_quad_gen_1 (d01sjc) is used and for $n > 3$ 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 References

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

5 Arguments

1: **tail** – Nag_TailProbability *Input*

On entry: indicates which probability is to be returned.

tail = Nag_LowerTail
The lower tail probability is returned.

tail = Nag_UpperTail
The upper tail probability is returned.

tail = Nag_Central
The central probability is returned.

Constraint: **tail** = Nag_LowerTail, Nag_UpperTail or Nag_Central.

2: **n** – Integer *Input*

On entry: n, the number of dimensions.

Constraint: $1 \leq$ **n** ≤ 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, **a**$[i-1] <$ **b**$[i-1]$, for $i = 1, 2, \dots, n$.

5: **mean**[**n**] – const double *Input*

On entry: μ, the mean vector of the multivariate Normal distribution.

6: **sigma**[**n** × **tdsig**] – const double *Input*

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

On entry: Σ, the variance-covariance matrix of the multivariate Normal distribution. Only the lower triangle is referenced.

Constraint: Σ must be positive definite.

7: **tdsig** – Integer *Input*

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

Constraint: **tdsig** $\geq$ **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 7.

If $n = 1$, **tol** is not referenced.

Suggested value: **tol** = 0.0001.

Constraint: if **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_1d_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 $n = 3$.

Constraint: if $\mathbf{n} \geq 3$, $\mathbf{maxpts} \geq 4 \times \mathbf{n}$.

10: **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, **tdsig** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: $\mathbf{tdsig} \geq \mathbf{n}$.

NE_2_REAL_ARRAYS_CONS

On entry, the ⟨*value*⟩ value in **b** is less than or equal to the corresponding value in **a**.

NE_ACC

Full accuracy not achieved, relative accuracy = ⟨*value*⟩. A larger value of **tol** can be tried or the length of the workspace increased. The returned value is an approximation to the required result.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT_ARG_CONS

On entry, **maxpts** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if $\mathbf{n} \geq 3$, $\mathbf{maxpts} \geq 4 \times \mathbf{n}$.

On entry, **n** = ⟨*value*⟩.
Constraint: $1 \leq \mathbf{n} \leq 10$.

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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

NE_POS_DEF

On entry, **sigma** is not positive definite.

NE_REAL_ARG_CONS

On entry, **tol** $= \langle value \rangle$.
Constraint: **tol** > 0.0.

NE_ROUND_OFF

Accuracy requested by **tol** is too strict: **tol** $= \langle value \rangle$. Round-off error has prevented the requested accuracy from being achieved; a larger value of **tol** can be tried. The returned value will be an approximation to the required result.

7 Accuracy

The accuracy should be as specified by **tol**. When on exit **fail.code** = NE_ACC the approximate accuracy achieved is given in the error message. 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 $\Phi^{-1}(\mathbf{tol}/(10 \times n))$, where Φ^{-1} is the inverse univariate Normal distribution function.

8 Parallelism and Performance

nag_multi_normal (g01hbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_multi_normal (g01hbc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

The time taken is related to the number of dimensions, the range over which the integration takes place ($b_i - a_i$, for $i = 1, 2, \ldots, n$) 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.

10 Example

This example reads in the mean and covariance matrix for a multivariate Normal distribution and computes and prints the associated central probability.

10.1 Program Text

```
/* nag_multi_normal (g01hbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

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

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

#define SIGMA(I, J) sigma[((I) -1)*n + (J) -1]
int main(void)
{
  Integer exit_status = 0, i, j, maxpts, n;
  char nag_enum_arg[40];
  double *a = 0, *b = 0, *mean = 0, prob, *sigma = 0, tol;
  Nag_TailProbability tail;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_multi_normal (g01hbc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT " %lf %39s", &n, &tol, nag_enum_arg,
          (unsigned)_countof(nag_enum_arg));
#else
  scanf("%" NAG_IFMT " %lf %39s", &n, &tol, nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  tail = (Nag_TailProbability) nag_enum_name_to_value(nag_enum_arg);

  if (!(a = NAG_ALLOC(n, double))
      || !(b = NAG_ALLOC(n, double))
      || !(mean = NAG_ALLOC(n, double))
      || !(sigma = NAG_ALLOC(n * n, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  for (j = 1; j <= n; ++j)
#ifdef _WIN32
    scanf_s("%lf", &mean[j - 1]);
#else
    scanf("%lf", &mean[j - 1]);
#endif
  for (i = 1; i <= n; ++i)
    for (j = 1; j <= n; ++j)
#ifdef _WIN32
      scanf_s("%lf", &SIGMA(i, j));
#else
      scanf("%lf", &SIGMA(i, j));
#endif
  if (tail == Nag_Central || tail == Nag_UpperTail)
    for (j = 1; j <= n; ++j)
#ifdef _WIN32
      scanf_s("%lf", &a[j - 1]);
#else
      scanf("%lf", &a[j - 1]);
#endif
  if (tail == Nag_Central || tail == Nag_LowerTail)
    for (j = 1; j <= n; ++j)
#ifdef _WIN32
      scanf_s("%lf", &b[j - 1]);
#else
      scanf("%lf", &b[j - 1]);
#endif
  maxpts = 2000;
```

```
  /* nag_multi_normal (g01hbc).
   * Computes probabilities for the multivariate Normal
   * distribution
   */
  prob = nag_multi_normal(tail, n, a, b, mean, sigma, n, tol, maxpts, &fail);
  if (fail.code == NE_NOERROR || fail.code == NE_ACC
      || fail.code == NE_ROUND_OFF) {
    printf("\nMultivariate Normal probability =  %6.4f\n", prob);
  }
  else {
    printf("Error from nag_multi_normal (g01hbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(mean);
  NAG_FREE(sigma);
  return exit_status;
}
```

10.2 Program Data

```
nag_multi_normal (g01hbc) Example Program Data
 4 0.0001 Nag_Central

 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
```

10.3 Program Results

```
nag_multi_normal (g01hbc) Example Program Results

Multivariate Normal probability =  0.9142
```