NAG Library Function Document

nag_deviates_gamma_vector (g01tfc)

1 Purpose

nag_deviates_gamma_vector (g01tfc) returns a number of deviates associated with given probabilities of the gamma distribution.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
void nag_deviates_gamma_vector (Integer ltail,
     const Nag_TailProbability tail[], Integer lp, const double p[],
     Integer la, const double a[], Integer lb, const double b[], double tol,
     double g[], Integer ivalid[], NagError *fail)
```

3 Description

The deviate, g_{p_i}, associated with the lower tail probability, p_i, of the gamma distribution with shape parameter α_i and scale parameter β_i, is defined as the solution to

$$P\left(G_i \leq g_{p_i} : \alpha_i, \beta_i\right) = p_i = \frac{1}{{\beta_i}^{\alpha_i}\Gamma(\alpha_i)}\int_0^{g_{p_i}} {e_i}^{-G_i/\beta_i} {G_i}^{\alpha_i - 1}\, dG_i, \quad 0 \leq g_{p_i} < \infty;\ \alpha_i, \beta_i > 0.$$

The method used is described by Best and Roberts (1975) making use of the relationship between the gamma distribution and the χ^2-distribution.

Let $y_i = 2\frac{g_{p_i}}{\beta_i}$. The required y_i is found from the Taylor series expansion

$$y_i = y_0 + \sum_r \frac{C_r(y_0)}{r!}\left(\frac{E_i}{\phi(y_0)}\right)^r,$$

where y_0 is a starting approximation

$$C_1(u_i) = 1,$$

$$C_{r+1}(u_i) = \left(r\Psi + \frac{d}{du_i}\right)C_r(u_i),$$

$$\Psi_i = \tfrac{1}{2} - \frac{\alpha_i - 1}{u_i},$$

$$E_i = p_i - \int_0^{y_0} \phi_i(u_i)\, du_i,$$

$$\phi_i(u_i) = \frac{1}{2^{\alpha_i}\Gamma(\alpha_i)} {e_i}^{-u_i/2} {u_i}^{\alpha_i - 1}.$$

For most values of p_i and α_i the starting value

$$y_{01} = 2\alpha_i\left(z_i\sqrt{\frac{1}{9\alpha_i}} + 1 - \frac{1}{9\alpha_i}\right)^3$$

is used, where z_i is the deviate associated with a lower tail probability of p_i for the standard Normal distribution.

For p_i close to zero,

$$y_{02} = (p_i \alpha_i 2^{\alpha_i} \Gamma(\alpha_i))^{1/\alpha_i}$$

is used.

For large p_i values, when $y_{01} > 4.4\alpha_i + 6.0$,

$$y_{03} = -2\left[\ln(1-p_i) - (\alpha_i - 1)\ln\left(\tfrac{1}{2}y_{01}\right) + \ln(\Gamma(\alpha_i))\right]$$

is found to be a better starting value than y_{01}.

For small α_i ($\alpha_i \leq 0.16$), p_i is expressed in terms of an approximation to the exponential integral and y_{04} is found by Newton–Raphson iterations.

Seven terms of the Taylor series are used to refine the starting approximation, repeating the process if necessary until the required accuracy is obtained.

The input arrays to this function are designed to allow maximum flexibility in the supply of vector arguments by re-using elements of any arrays that are shorter than the total number of evaluations required. See Section 2.6 in the g01 Chapter Introduction for further information.

4 References

Best D J and Roberts D E (1975) Algorithm AS 91. The percentage points of the χ^2 distribution *Appl. Statist.* **24** 385–388

5 Arguments

1: **ltail** – Integer *Input*

On entry: the length of the array **tail**.

Constraint: **ltail** > 0.

2: **tail**[**ltail**] – const Nag_TailProbability *Input*

On entry: indicates which tail the supplied probabilities represent. For $j = (i-1)$ mod **ltail**, for $i = 1, 2, \ldots, \max(\textbf{ltail}, \textbf{lp}, \textbf{la}, \textbf{lb})$:

tail[j] = Nag_LowerTail
The lower tail probability, i.e., $p_i = P\left(G_i \leq g_{p_i} : \alpha_i, \beta_i\right)$.

tail[j] = Nag_UpperTail
The upper tail probability, i.e., $p_i = P\left(G_i \geq g_{p_i} : \alpha_i, \beta_i\right)$.

Constraint: **tail**[$j-1$] = Nag_LowerTail or Nag_UpperTail, for $j = 1, 2, \ldots,$ **ltail**.

3: **lp** – Integer *Input*

On entry: the length of the array **p**.

Constraint: **lp** > 0.

4: **p**[**lp**] – const double *Input*

On entry: p_i, the probability of the required gamma distribution as defined by **tail** with $p_i =$ **p**[j], $j = (i-1)$ mod **lp**.

Constraints:

if **tail**[k] = Nag_LowerTail, $0.0 \leq$ **p**[j] < 1.0;
otherwise $0.0 <$ **p**[j] ≤ 1.0.

Where $k = (i-1)$ mod **ltail** and $j = (i-1)$ mod **lp**.

5: **la** – Integer *Input*

On entry: the length of the array **a**.

Constraint: **la** > 0.

6: **a**[**la**] – const double *Input*

On entry: α_i, the first parameter of the required gamma distribution with $\alpha_i = \mathbf{a}[j]$, $j = (i-1) \bmod \mathbf{la}$.

Constraint: $0.0 < \mathbf{a}[j-1] \leq 10^6$, for $j = 1, 2, \ldots, \mathbf{la}$.

7: **lb** – Integer *Input*

On entry: the length of the array **b**.

Constraint: **lb** > 0.

8: **b**[**lb**] – const double *Input*

On entry: β_i, the second parameter of the required gamma distribution with $\beta_i = \mathbf{b}[j]$, $j = (i-1) \bmod \mathbf{lb}$.

Constraint: $\mathbf{b}[j-1] > 0.0$, for $j = 1, 2, \ldots, \mathbf{lb}$.

9: **tol** – double *Input*

On entry: the relative accuracy required by you in the results. If nag_deviates_gamma_vector (g01tfc) is entered with **tol** greater than or equal to 1.0 or less than $10 \times$ ***machine precision*** (see nag_machine_precision (X02AJC)), then the value of $10 \times$ ***machine precision*** is used instead.

10: **g**[dim] – double *Output*

Note: the dimension, *dim*, of the array **g** must be at least max(**ltail**, **lp**, **la**, **lb**).

On exit: g_{p_i}, the deviates for the gamma distribution.

11: **ivalid**[dim] – Integer *Output*

Note: the dimension, *dim*, of the array **ivalid** must be at least max(**ltail**, **lp**, **la**, **lb**).

On exit: **ivalid**$[i-1]$ indicates any errors with the input arguments, with

ivalid$[i-1] = 0$
No error.

ivalid$[i-1] = 1$
On entry, invalid value supplied in **tail** when calculating g_{p_i}.

ivalid$[i-1] = 2$
On entry, invalid value for p_i.

ivalid$[i-1] = 3$
On entry, $\alpha_i \leq 0.0$,
or $\alpha_i > 10^6$,
or $\beta_i \leq 0.0$.

ivalid$[i-1] = 4$
p_i is too close to 0.0 or 1.0 to enable the result to be calculated.

ivalid$[i-1] = 5$
The solution has failed to converge. The result may be a reasonable approximation.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_ARRAY_SIZE

On entry, array size $= \langle value \rangle$.
Constraint: **la** > 0.

On entry, array size $= \langle value \rangle$.
Constraint: **lb** > 0.

On entry, array size $= \langle value \rangle$.
Constraint: **lp** > 0.

On entry, array size $= \langle value \rangle$.
Constraint: **ltail** > 0.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

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 the Essential Introduction 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 the Essential Introduction for further information.

NW_IVALID

On entry, at least one value of **tail**, **p**, **a**, or **b** was invalid.
Check **ivalid** for more information.

7 Accuracy

In most cases the relative accuracy of the results should be as specified by **tol**. However, for very small values of α_i or very small values of p_i there may be some loss of accuracy.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example reads lower tail probabilities for several gamma distributions, and calculates and prints the corresponding deviates until the end of data is reached.

10.1 Program Text

```
/* nag_deviates_gamma_vector (g01tfc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
  /* Integer scalar and array declarations */
  Integer ltail, lp, la, lb, i, lout;
  Integer *ivalid = 0;
  Integer exit_status = 0;

  /* NAG structures */
  NagError fail;
  Nag_TailProbability *tail = 0;

  /* Double scalar and array declarations */
  double tol;
  double *p = 0, *a = 0, *b = 0, *g = 0;

  /* Character scalar and array declarations */
  char ctail[40];

  /* Initialise the error structure to print out any error messages */
  INIT_FAIL(fail);

  printf("nag_deviates_gamma_vector (g01tfc) Example Program Results\n\n");

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

  /* Read in the tolerance */
#ifdef _WIN32
  scanf_s("%lf%*[^\n] ", &tol);
#else
  scanf("%lf%*[^\n] ", &tol);
#endif

  /* Read in the input vectors */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &ltail);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &ltail);
#endif
  if (!(tail = NAG_ALLOC(ltail, Nag_TailProbability))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ltail; i++) {
#ifdef _WIN32
    scanf_s("%39s", ctail, _countof(ctail));
```

```
#else
    scanf("%39s", ctail);
#endif
    tail[i] = (Nag_TailProbability) nag_enum_name_to_value(ctail);
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &lp);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &lp);
#endif
  if (!(p = NAG_ALLOC(lp, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lp; i++)
#ifdef _WIN32
    scanf_s("%lf", &p[i]);
#else
    scanf("%lf", &p[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &la);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &la);
#endif
  if (!(a = NAG_ALLOC(la, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < la; i++)
#ifdef _WIN32
    scanf_s("%lf", &a[i]);
#else
    scanf("%lf", &a[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &lb);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &lb);
#endif
  if (!(b = NAG_ALLOC(lb, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lb; i++)
#ifdef _WIN32
    scanf_s("%lf", &b[i]);
#else
    scanf("%lf", &b[i]);
#endif
```

```
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Allocate memory for output */
  lout = MAX(ltail,MAX(lp,MAX(la,lb)));
  if (!(g = NAG_ALLOC(lout, double)) ||
      !(ivalid = NAG_ALLOC(lout, Integer))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Calculate probability */
  nag_deviates_gamma_vector(ltail, tail, lp, p, la, a, lb, b, tol, g,
                            ivalid, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_deviates_gamma_vector (g01tfc).\n%s\n",
            fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID) goto END;
  }

  /* Display title */
  printf("        tail         p          a         b        g     ivalid\n");
  printf(" ---------------------------------------------------------------\n");

  /* Display results */
  for (i = 0; i < lout; i++)
    printf(" %15s  %6.3f    %6.2f    %6.2f   %7.3f    %3"NAG_IFMT"\n",
           nag_enum_value_to_name(tail[i%ltail]),p[i%lp], a[i%la],
           b[i%lb], g[i], ivalid[i]);

 END:
  NAG_FREE(tail);
  NAG_FREE(p);
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(g);
  NAG_FREE(ivalid);

  return(exit_status);
}
```

10.2 Program Data

```
nag_deviates_gamma_vector (g01tfc) Example Program Data
0.0                                                                    :: tol
1                                                                      :: ltail
Nag_LowerTail                                                          :: tail
3                                                                      :: lp
0.01 0.428 0.869                                                       :: p
3                                                                      :: la
1.0 7.5 45.0                                                           :: a
3                                                                      :: lb
20.0 0.1 10.0                                                          :: b
```

10.3 Program Results

```
nag_deviates_gamma_vector (g01tfc) Example Program Results

        tail         p          a         b        g     ivalid
 ---------------------------------------------------------------
   Nag_LowerTail   0.010      1.00     20.00     0.201      0
   Nag_LowerTail   0.428      7.50      0.10     0.670      0
   Nag_LowerTail   0.869     45.00     10.00   525.839      0
```