NAG Library Function Document

nag_prob_non_central_beta_dist (g01gec)

1 Purpose

nag_prob_non_central_beta_dist (g01gec) returns the probability associated with the lower tail of the noncentral beta distribution.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
double nag_prob_non_central_beta_dist (double x, double a, double b,
       double lambda, double tol, Integer max_iter, NagError *fail)
```

3 Description

The lower tail probability for the noncentral beta distribution with parameters a and b and noncentrality parameter λ, $P(B \leq \beta : a, b; \lambda)$, is defined by

$$P(B \leq \beta : a, b; \lambda) = \sum_{j=0}^{\infty} e^{-\lambda/2} \frac{(\lambda/2)}{j!} P(B \leq \beta : a, b; 0), \tag{1}$$

where

$$P(B \leq \beta : a, b; 0) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^{\beta} B^{a-1}(1-B)^{b-1}\, dB,$$

which is the central beta probability function or incomplete beta function.

Recurrence relationships given in Abramowitz and Stegun (1972) are used to compute the values of $P(B \leq \beta : a, b; 0)$ for each step of the summation (1).

The algorithm is discussed in Lenth (1987).

4 References

Abramowitz M and Stegun I A (1972) *Handbook of Mathematical Functions* (3rd Edition) Dover Publications

Lenth R V (1987) Algorithm AS 226: Computing noncentral beta probabilities *Appl. Statist.* **36** 241–244

5 Arguments

1: **x** – double *Input*

On entry: β, the deviate from the beta distribution, for which the probability $P(B \leq \beta : a, b; \lambda)$ is to be found.

Constraint: $0.0 \leq \mathbf{x} \leq 1.0$.

2: **a** – double *Input*

On entry: a, the first parameter of the required beta distribution.

Constraint: $0.0 < \mathbf{a} \leq 10^6$.

3: **b** – double *Input*

On entry: b, the second parameter of the required beta distribution.

Constraint: $0.0 < \mathbf{b} \leq 10^6$.

4: **lambda** – double *Input*

On entry: λ, the noncentrality parameter of the required beta distribution.

Constraint: $0.0 \leq \mathbf{lambda} \leq -2.0\log(U)$, where U is the safe range parameter as defined by nag_real_safe_small_number (X02AMC).

5: **tol** – double *Input*

On entry: the relative accuracy required by you in the results. If nag_prob_non_central_beta_dist (g01gec) 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.

See Section 7 for the relationship between **tol** and **max_iter**.

6: **max_iter** – Integer *Input*

On entry: the maximum number of iterations that the algorithm should use.

See Section 7 for suggestions as to suitable values for **max_iter** for different values of the arguments.

Suggested value: 500.

Constraint: **max_iter** ≥ 1.

7: **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_CONV

The solution has failed to converge in $\langle value \rangle$ iterations. Consider increasing **max_iter** or **tol**.

NE_INT_ARG_LT

On entry, **max_iter** $= \langle value \rangle$.
Constraint: **max_iter** ≥ 1.

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.

NE_PROB_B_INIT

The required accuracy was not achieved when calculating the initial value of the beta distribution. You should try a larger value of **tol**. The returned value will be an approximation to the correct value. This error exit is no longer possible but is still documented for historical reasons.

NE_PROB_LIMIT

The probability is too close to 0.0 or 1.0 for the algorithm to be able to calculate the required probability. nag_prob_non_central_beta_dist (g01gec) will return 0.0 or 1.0 as appropriate. This should be a reasonable approximation.

NE_REAL_ARG_CONS

On entry, $\mathbf{a} = \langle value \rangle$.
Constraint: $0.0 < \mathbf{a} \leq 10^6$.

On entry, $\mathbf{b} = \langle value \rangle$.
Constraint: $0.0 < \mathbf{b} \leq 10^6$.

On entry, **lambda** $= \langle value \rangle$.
Constraint: $0.0 \leq$ **lambda** $\leq -2.0\log(U)$, where U is the safe range parameter as defined by nag_real_safe_small_number (X02AMC).

On entry, $\mathbf{x} = \langle value \rangle$.
Constraint: $0.0 \leq \mathbf{x} \leq 1.0$.

7 Accuracy

Convergence is theoretically guaranteed whenever $P(Y >$ **max_iter**$) \leq$ **tol** where Y has a Poisson distribution with mean $\lambda/2$. Excessive round-off errors are possible when the number of iterations used is high and **tol** is close to ***machine precision***. See Lenth (1987) for further comments on the error bound.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The central beta probabilities can be obtained by setting **lambda** $= 0.0$.

10 Example

This example reads values for several beta distributions and calculates and prints the lower tail probabilities until the end of data is reached.

10.1 Program Text

```
/* nag_prob_non_central_beta_dist (g01gec) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 */

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


int main(void)
{
  Integer  exit_status = 0, max_iter;
```

```
  NagError fail;
  double   a, b, lambda, prob, tol, x;

  INIT_FAIL(fail);

  printf(
          "nag_prob_non_central_beta_dist (g01gec) Example Program Results\n");

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

  printf("\n     x        a       b       lambda   prob\n\n");
  tol = 5e-6;
  max_iter = 50;
#ifdef _WIN32
  while ((scanf_s("%lf %lf %lf %lf %*[^\n]", &x, &a, &b, &lambda)) != EOF)
#else
  while ((scanf("%lf %lf %lf %lf %*[^\n]", &x, &a, &b, &lambda)) != EOF)
#endif
    {
      /* nag_prob_non_central_beta_dist (g01gec).
       * Computes probabilities for the non-central beta
       * distribution
       */
      prob = nag_prob_non_central_beta_dist(x, a, b, lambda, tol, max_iter,
                                            &fail);
      if (fail.code != NE_NOERROR)
        {
          printf(
                  "Error from nag_prob_non_central_beta_dist (g01gec).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      printf("%8.3f %8.3f %8.3f %8.3f %8.4f\n", x, a, b, lambda, prob);
    }
 END:
  return exit_status;
}
```

10.2 Program Data

```
nag_prob_non_central_beta_dist (g01gec) Example Program Data
 0.25  1.0  2.0  1.0         :x a lambda
 0.75  1.5  1.5  0.5         :x a lambda
 0.5   2.0  1.0  0.0         :x a lambda
```

10.3 Program Results

```
nag_prob_non_central_beta_dist (g01gec) Example Program Results

     x        a       b       lambda   prob

   0.250    1.000    2.000    1.000   0.3168
   0.750    1.500    1.500    0.500   0.7705
   0.500    2.000    1.000    0.000   0.2500
```