NAG Library Function Document

nag_deviates_beta (g01fec)

1 Purpose

nag_deviates_beta (g01fec) returns the deviate associated with the given lower tail probability of the beta distribution.

2 Specification

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

```
double nag_deviates_beta (double p, double a, double b, double tol,
       NagError *fail)
```

3 Description

The deviate, β_p, associated with the lower tail probability, p, of the beta distribution with parameters a and b is defined as the solution to

$$P\big(B \leq \beta_p : a, b\big) = p = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^{\beta_p} B^{a-1}(1-B)^{b-1}\, dB, \quad 0 \leq \beta_p \leq 1; a, b > 0.$$

The algorithm is a modified version of the Newton–Raphson method, following closely that of Cran *et al.* (1977).

An initial approximation, β_0, to β_p is found (see Cran *et al.* (1977)), and the Newton–Raphson iteration

$$\beta_i = \beta_{i-1} - \frac{f(\beta_{i-1})}{f'(\beta_{i-1})},$$

where $f(\beta) = P(B \leq \beta : a, b) - p$ is used, with modifications to ensure that β remains in the range $(0, 1)$.

4 References

Cran G W, Martin K J and Thomas G E (1977) Algorithm AS 109. Inverse of the incomplete beta function ratio *Appl. Statist.* **26** 111–114

Hastings N A J and Peacock J B (1975) *Statistical Distributions* Butterworth

5 Arguments

1: **p** – double *Input*

On entry: p, the lower tail probability from the required beta distribution.

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

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: **tol** – double *Input*

On entry: the relative accuracy required by you in the result. If nag_deviates_beta (g01fec) 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.

5: **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

On any of the error conditions listed below except **fail**.**code** = NE_RES_NOT_ACC or NE_SOL_NOT_CONV nag_deviates_beta (g01fec) returns 0.0.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

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 2.7.6 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 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_REAL_ARG_GT

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

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

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

NE_REAL_ARG_LE

On entry, $\mathbf{a} = \langle\mathit{value}\rangle$ and $\mathbf{b} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{a} > 0.0$.

On entry, $\mathbf{a} = \langle\mathit{value}\rangle$ and $\mathbf{b} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{b} > 0.0$.

NE_REAL_ARG_LT

On entry, $\mathbf{p} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{p} \geq 0.0$.

NE_RES_NOT_ACC

The requested accuracy has not been achieved. Use a larger value of **tol**. There is doubt concerning the accuracy of the computed result. 100 iterations of the Newton–Raphson method have been performed without satisfying the accuracy criterion (see Section 9). The result should be a reasonable approximation of the solution.

NE_SOL_NOT_CONV

The solution has failed to converge. However, the result should be a reasonable approximation. Requested accuracy not achieved when calculating beta probability. You should try setting **tol** larger.

7 Accuracy

The required precision, given by **tol**, should be achieved in most circumstances.

8 Parallelism and Performance

nag_deviates_beta (g01fec) is not threaded in any implementation.

9 Further Comments

The typical timing will be several times that of nag_prob_beta_dist (g01eec) and will be very dependent on the input argument values. See nag_prob_beta_dist (g01eec) for further comments on timings.

10 Example

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

10.1 Program Text

```
/* nag_deviates_beta (g01fec) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
  Integer exit_status = 0;
  double a, b, p, tol, x;
  NagError fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  printf("nag_deviates_beta (g01fec) Example Program Results\n");
  printf(" Probability     A         B     Deviate\n\n");
#ifdef _WIN32
```

```
  while (scanf_s("%lf %lf %lf", &p, &a, &b) != EOF)
#else
  while (scanf("%lf %lf %lf", &p, &a, &b) != EOF)
#endif
  {
    tol = 0.0;
    /* nag_deviates_beta (g01fec).
     * Deviates for the beta distribution
     */
    x = nag_deviates_beta(p, a, b, tol, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_deviates_beta (g01fec).\n%s\n", fail.message);
      exit_status = 1;
      if (fail.code != NE_RES_NOT_ACC && fail.code != NE_SOL_NOT_CONV) {
        goto END;
      }
    }
    printf("%9.4f%10.3f%10.3f%10.4f\n", p, a, b, x);
  }

END:
  return exit_status;
}
```

10.2 Program Data

```
nag_deviates_beta (g01fec) Example Program Data
0.5000  1.0  2.0
0.9900  1.5  1.5
0.2500 20.0 10.0
```

10.3 Program Results

```
nag_deviates_beta (g01fec) Example Program Results
 Probability     A         B     Deviate

   0.5000     1.000     2.000    0.2929
   0.9900     1.500     1.500    0.9672
   0.2500    20.000    10.000    0.6105
```