nag_gamma (s14aac)

1. Purpose

nag_gamma (s14aac) returns the value of the Gamma function $\Gamma(x)$.

2. Specification

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

double nag_gamma(double x, NagError *fail)
```

3. Description

This function evaluates

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt.$$

The function is based on a Chebyshev expansion for $\Gamma(1+u)$, and uses the property $\Gamma(1+x) = x\Gamma(x)$. If $x = N + 1 + u$ where N is integral and $0 \leq u < 1$ then it follows that:

for $N > 0$ $\quad \Gamma(x) = (x-1)(x-2)\ldots(x-N)\Gamma(1+u)$
for $N = 0$ $\quad \Gamma(x) = \Gamma(1+u)$
for $N < 0$ $\quad \Gamma(x) = \Gamma(1+u)/x(x+1)(x+2)\ldots(x-N-1)$.

There are four possible failures for this function:

(i) if x is too large, there is a danger of overflow since $\Gamma(x)$ could become too large to be represented in the machine;
(ii) if x is too large and negative, there is a danger of underflow;
(iii) if x is equal to a negative integer, $\Gamma(x)$ would overflow since it has poles at such points;
(iv) if x is too near zero, there is again the danger of overflow on some machines.

For small x, $\Gamma(x) \simeq 1/x$, and on some machines there exists a range of non-zero but small values of x for which $1/x$ is larger than the greatest representable value.

4. Parameters

x
Input: the argument x of the function.
Constraint: **x** must not be zero or a negative integer.

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_REAL_ARG_GT
On entry, **x** must not be greater than ⟨*value*⟩: **x** = ⟨*value*⟩.
The argument is too large, the function returns the approximate value of $\Gamma(x)$ at the nearest valid argument.

NE_REAL_ARG_LT
On entry, **x** must not be less than ⟨*value*⟩: **x** = ⟨*value*⟩.
The argument is too large and negative, the function returns zero.

NE_REAL_ARG_TOO_SMALL
On entry, **x** must be greater than ⟨*value*⟩: **x** = ⟨*value*⟩.
The argument is too close to zero, the function returns the approximate value of $\Gamma(x)$ at the nearest valid argument.

NE_REAL_ARG_NEG_INT
On entry, **x** must not be effectively a negative integer: **x** = ⟨*value*⟩.
The argument is a negative integer, at which values $\Gamma(x)$ is infinite. The function returns a large positive value.

6. Further Comments

6.1. Accuracy

Let δ and ϵ be the relative errors in the argument and the result respectively. If δ is somewhat larger than the ***machine precision*** (i.e., is due to data errors etc.), then ϵ and δ are approximately related by $\epsilon \simeq |x\psi(x)|\,\delta$ (provided ϵ is also greater than the representation error). Here $\psi(x)$ is the digamma function $\Gamma'(x)/\Gamma(x)$.

If δ is of the same order as ***machine precision***, then rounding errors could make ϵ slightly larger than the above relation predicts.

There is clearly a severe, but unavoidable, loss of accuracy for arguments close to the poles of $\Gamma(x)$ at negative integers. However, relative accuracy is preserved near the pole at $x = 0$ right up to the point of failure arising from the danger of setting overflow.

Also accuracy will necessarily be lost as x becomes large since in this region $\epsilon \simeq \delta x \ln x$. However, since $\Gamma(x)$ increases rapidly with x, the function must fail due to the danger of setting overflow before this loss of accuracy is too great. For example, for $x = 20$, the amplification factor $\simeq 60$.

6.2. References

Abramowitz M and Stegun I A (1968) *Handbook of Mathematical Functions* Dover Publications, New York ch 6 p 255.

7. See Also

nag_log_gamma (s14abc)
nag_incomplete_gamma (s14bac)

8. Example

The following program reads values of the argument x from a file, evaluates the function at each value of x and prints the results.

8.1. Program Text

```
/* nag_gamma(s14aac) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 2 revised, 1992.
 */

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

main()
{
  double x, y;

  /* Skip heading in data file */
  Vscanf("%*[^\n]");
  Vprintf("s14aac Example Program Results\n");
  Vprintf("     x           y\n");
  while (scanf("%lf", &x) != EOF)
    {
      y = s14aac(x, NAGERR_DEFAULT);
      Vprintf("%12.3e%12.3e\n", x, y);
    }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
s14aac Example Program Data
             1.0
             1.25
             1.5
             1.75
             2.0
             5.0
             10.0
             -1.5
```

8.3. Program Results

```
s14aac Example Program Results
     x           y
   1.000e+00   1.000e+00
   1.250e+00   9.064e-01
   1.500e+00   8.862e-01
   1.750e+00   9.191e-01
   2.000e+00   1.000e+00
   5.000e+00   2.400e+01
   1.000e+01   3.629e+05
  -1.500e+00   2.363e+00
```