NAG Library Function Document

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

nag_gamma (s14aac) evaluates an approximation to the gamma function $\Gamma(x)$. The function is based on the Chebyshev expansion:

$$\Gamma(1+u) = \sum_{r=0}^{\prime} a_r T_r(t), \qquad \text{where } 0 \leq u < 1, t = 2u - 1,$$

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 $N > 0$, $\quad \Gamma(x) = (x-1)(x-2)\cdots(x-N)\Gamma(1+u)$,

for $N = 0$, $\quad \Gamma(x) = \Gamma(1+u)$,

for $N < 0$, $\quad \Gamma(x) = \dfrac{\Gamma(1+u)}{x(x+1)(x+2)\cdots(x-N-1)}$.

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 nonzero but small values of x for which $1/x$ is larger than the greatest representable value.

4 References

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

5 Arguments

1: **x** – double *Input*

On entry: the argument x of the function.

Constraint: **x** must not be zero or a negative integer.

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

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

6 Error Indicators and Warnings

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.

NE_REAL_ARG_GT

On entry, **x** = ⟨*value*⟩.
Constraint: **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** = ⟨*value*⟩. The function returns zero.
Constraint: **x** ≥ ⟨*value*⟩.
The argument is too large and negative, the function returns zero.

NE_REAL_ARG_NEG_INT

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

NE_REAL_ARG_TOO_SMALL

On entry, **x** = ⟨*value*⟩.
Constraint: |**x**| ≥ ⟨*value*⟩.
The argument is too close to zero, the function returns the approximate value of $\Gamma(x)$ at the nearest valid argument.

7 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 $\frac{\Gamma'(x)}{\Gamma(x)}$. Figure 1 shows the behaviour of the error amplification factor $|x\Psi(x)|$.

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 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 overflow before this loss of accuracy is too great. (For example, for $x = 20$, the amplification factor $\simeq 60$.)

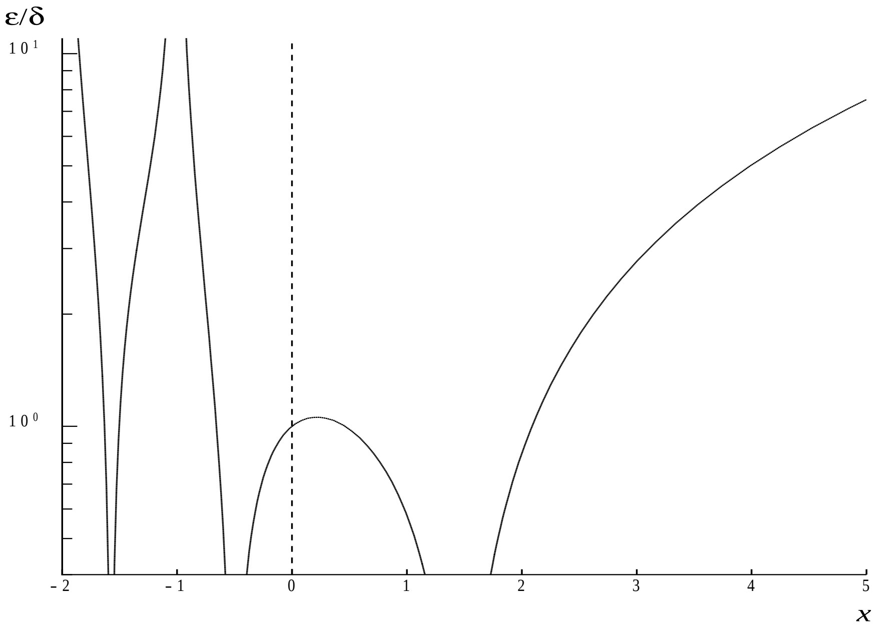


Figure 1

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example reads values of the argument x from a file, evaluates the function at each value of x and prints the results.

10.1 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>

int main(void)
{
  Integer  exit_status = 0;
  double   x, y;
  NagError fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
  scanf("%*[^\n]");
  printf("nag_gamma (s14aac) Example Program Results\n");
  printf("     x           y\n");
```

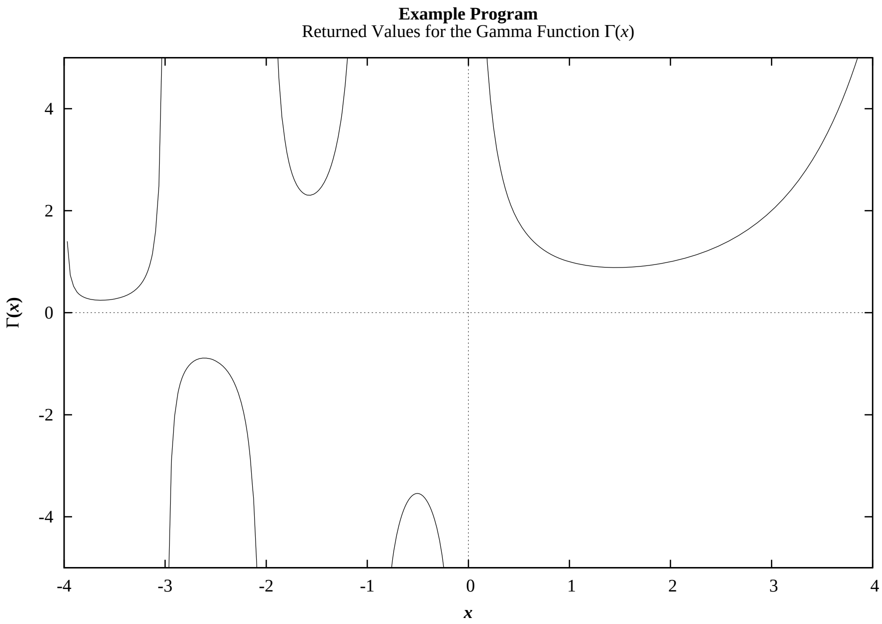
```
  while (scanf("%lf", &x) != EOF)
    {
      /* nag_gamma (s14aac).
       * Gamma function Gamma(x)
       */
      y = nag_gamma(x, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_gamma (s14aac).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      printf("%12.3e%12.3e\n", x, y);
    }

 END:
  return exit_status;
}
```

10.2 Program Data

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

10.3 Program Results

```
nag_gamma (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
```


Example Program
Returned Values for the Gamma Function Γ(x)
4
2
0
-2
-4
Γ(x)
-4
-3
-2
-1
0
1
2
3
4
x