Hart, J.F., et al. 1968, *Computer Approximations* (New York: Wiley).
Hastings, C. 1955, *Approximations for Digital Computers* (Princeton: Princeton University Press).
Luke, Y.L. 1975, *Mathematical Functions and Their Approximations* (New York: Academic Press).

## 6.1 Gamma Function, Beta Function, Factorials, Binomial Coefficients

The gamma function is defined by the integral

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{6.1.1}$$

When the argument $z$ is an integer, the gamma function is just the familiar factorial function, but offset by one,

$$n! = \Gamma(n+1) \tag{6.1.2}$$

The gamma function satisfies the recurrence relation

$$\Gamma(z+1) = z\Gamma(z) \tag{6.1.3}$$

If the function is known for arguments $z > 1$ or, more generally, in the half complex plane $\text{Re}(z) > 1$ it can be obtained for $z < 1$ or $\text{Re}\,(z) < 1$ by the reflection formula

$$\Gamma(1-z) = \frac{\pi}{\Gamma(z)\sin(\pi z)} = \frac{\pi z}{\Gamma(1+z)\sin(\pi z)} \tag{6.1.4}$$

Notice that $\Gamma(z)$ has a pole at $z = 0$, and at all negative integer values of $z$.

There are a variety of methods in use for calculating the function $\Gamma(z)$ numerically, but none is quite as neat as the approximation derived by Lanczos [1]. This scheme is entirely specific to the gamma function, seemingly plucked from thin air. We will not attempt to derive the approximation, but only state the resulting formula: For certain integer choices of $\gamma$ and $N$, and for certain coefficients $c_1, c_2, \ldots, c_N$, the gamma function is given by

$$\begin{aligned}\Gamma(z+1) &= (z+\gamma+\tfrac{1}{2})^{z+\frac{1}{2}} e^{-(z+\gamma+\frac{1}{2})} \\ &\quad\times \sqrt{2\pi}\left[c_0 + \frac{c_1}{z+1} + \frac{c_2}{z+2} + \cdots + \frac{c_N}{z+N} + \epsilon\right] \quad (z > 0)\end{aligned} \tag{6.1.5}$$

You can see that this is a sort of take-off on Stirling's approximation, but with a series of corrections that take into account the first few poles in the left complex plane. The constant $c_0$ is very nearly equal to 1. The error term is parametrized by $\epsilon$. For $\gamma = 5$, $N = 6$, and a certain set of $c$'s, the error is smaller than $|\epsilon| < 2 \times 10^{-10}$. Impressed? If not, then perhaps you will be impressed by the fact that (with these same parameters) the formula (6.1.5) and bound on $\epsilon$ apply for the *complex* gamma function, *everywhere in the half complex plane Re* $z > 0$.

It is better to implement $\ln\Gamma(x)$ than $\Gamma(x)$, since the latter will overflow many computers' floating-point representation at quite modest values of $x$. Often the gamma function is used in calculations where the large values of $\Gamma(x)$ are divided by other large numbers, with the result being a perfectly ordinary value. Such operations would normally be coded as subtraction of logarithms. With (6.1.5) in hand, we can compute the logarithm of the gamma function with two calls to a logarithm and 25 or so arithmetic operations. This makes it not much more difficult than other built-in functions that we take for granted, such as $\sin x$ or $e^x$:

```
      FUNCTION gammln(xx)
      REAL gammln,xx
         Returns the value ln[Γ(xx)] for xx > 0.
      INTEGER j
      DOUBLE PRECISION ser,stp,tmp,x,y,cof(6)
         Internal arithmetic will be done in double precision, a nicety that you can omit if five-figure
         accuracy is good enough.
      SAVE cof,stp
      DATA cof,stp/76.18009172947146d0,-86.50532032941677d0,
     *    24.01409824083091d0,-1.231739572450155d0,.1208650973866179d-2,
     *    -.5395239384953d-5,2.5066282746310005d0/
      x=xx
      y=x
      tmp=x+5.5d0
      tmp=(x+0.5d0)*log(tmp)-tmp
      ser=1.000000000190015d0
      do 11 j=1,6
          y=y+1.d0
          ser=ser+cof(j)/y
      enddo 11
      gammln=tmp+log(stp*ser/x)
      return
      END
```

How shall we write a routine for the factorial function $n!$? Generally the factorial function will be called for small integer values (for large values it will overflow anyway!), and in most applications the same integer value will be called for many times. It is a profligate waste of computer time to call `exp(gammln(n+1.0))` for each required factorial. Better to go back to basics, holding `gammln` in reserve for unlikely calls:

```
      FUNCTION factrl(n)
      INTEGER n
      REAL factrl
C     USES gammln
         Returns the value n! as a floating-point number.
      INTEGER j,ntop
      REAL a(33),gammln              Table to be filled in only as required.
      SAVE ntop,a
      DATA ntop,a(1)/0,1./           Table initialized with 0! only.
      if (n.lt.0) then
          pause 'negative factorial in factrl'
      else if (n.le.ntop) then       Already in table.
          factrl=a(n+1)
      else if (n.le.32) then         Fill in table up to desired value.
          do 11 j=ntop+1,n
```

```
        a(j+1)=j*a(j)
    enddo 11
    ntop=n
    factrl=a(n+1)
else                              Larger value than size of table is required. Actually, this big
    factrl=exp(gammln(n+1.))         a value is going to overflow on many computers, but no
endif                                harm in trying.
return
END
```

A useful point is that `factrl` will be *exact* for the smaller values of `n`, since floating-point multiplies on small integers are exact on all computers. This exactness will not hold if we turn to the logarithm of the factorials. For binomial coefficients, however, we must do exactly this, since the individual factorials in a binomial coefficient will overflow long before the coefficient itself will.

The binomial coefficient is defined by

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \qquad 0 \le k \le n \tag{6.1.6}$$

```
      FUNCTION bico(n,k)
      INTEGER k,n
      REAL bico
C     USES factln
         Returns the binomial coefficient (n k) as a floating-point number.
      REAL factln
      bico=nint(exp(factln(n)-factln(k)-factln(n-k)))
      return  The nearest-integer function cleans up roundoff error for smaller values of n and k.
      END
```

which uses

```
      FUNCTION factln(n)
      INTEGER n
      REAL factln
C     USES gammln
         Returns ln(n!).
      REAL a(100),gammln
      SAVE a
      DATA a/100*-1./                          Initialize the table to negative values.
      if (n.lt.0) pause 'negative factorial in factln'
      if (n.le.99) then                        In range of the table.
          if (a(n+1).lt.0.) a(n+1)=gammln(n+1.)   If not already in the table, put it in.
          factln=a(n+1)
      else
          factln=gammln(n+1.)                  Out of range of the table.
      endif
      return
      END
```

If your problem requires a series of related binomial coefficients, a good idea is to use recurrence relations, for example

$$\begin{aligned} \binom{n+1}{k} &= \frac{n+1}{n-k+1}\binom{n}{k} = \binom{n}{k} + \binom{n}{k-1} \\ \binom{n}{k+1} &= \frac{n-k}{k+1}\binom{n}{k} \end{aligned} \tag{6.1.7}$$

Finally, turning away from the combinatorial functions with integer valued arguments, we come to the beta function,

$$B(z,w) = B(w,z) = \int_0^1 t^{z-1}(1-t)^{w-1}dt \tag{6.1.8}$$

which is related to the gamma function by

$$B(z,w) = \frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)} \tag{6.1.9}$$

hence

```
      FUNCTION beta(z,w)
      REAL beta,w,z
C     USES gammln
         Returns the value of the beta function B(z, w).
      REAL gammln
      beta=exp(gammln(z)+gammln(w)-gammln(z+w))
      return
      END
```

CITED REFERENCES AND FURTHER READING:

Abramowitz, M., and Stegun, I.A. 1964, *Handbook of Mathematical Functions*, Applied Mathematics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 6.

Lanczos, C. 1964, *SIAM Journal on Numerical Analysis*, ser. B, vol. 1, pp. 86–96. [1]

# 6.2 Incomplete Gamma Function, Error Function, Chi-Square Probability Function, Cumulative Poisson Function

The incomplete gamma function is defined by

$$P(a,x) \equiv \frac{\gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)}\int_0^x e^{-t}t^{a-1}dt \qquad (a>0) \tag{6.2.1}$$