Besselk

`Besselk(n,z)`

The Besselk function returns the modified Bessel function of the second kind, K*n*(*z*), of order *n* and argument *z*. Replaces the bessK function, which is supported for backwards compatibility only.

If *z* is real, Besselk returns a real value, which means that if *z* is also negative, it returns NaN unless *n* is an integer.

For complex *z* a complex value is returned, and there are no restrictions on *z* except for possible overflow.

Details

The calculation is performed using the SLATEC library. The function supports fractional orders *n*, as well as real or complex arguments *z*.

See Also

The **Besseli**, **Besselj**, and **Bessely** functions.

Bessely

`Bessely(n,z)`

The Bessely function returns the Bessel function of the second kind, Y*n*(*z*), of order *n* and argument *z*. Replaces the bessY function, which is supported for backwards compatibility only.

If *z* is real, Bessely returns a real value, which means that if *z* is also negative, it returns NaN unless *n* is an integer.

For complex *z* a complex value is returned, and there are no restrictions on *z* except for possible overflow.

Details

The calculation is performed using the SLATEC library. The function supports fractional and negative orders *n*, as well as real or complex arguments *z*.

See Also

The **Besseli**, **Besselj**, and **Besselk** functions.

bessI

`bessI(n, x [, algorithm [, accuracy]])`

Obsolete — use **Besseli**.

The bessI function returns the modified Bessel function of the first kind, I*n*(*x*) of order *n* and argument *x*.

For real *x*, the optional parameter *algorithm* selects between a faster, less accurate calculation method and slower, more accurate methods. In addition, when *algorithm* is zero or absent, the order *n* is truncated to an integer.

When *algorithm* is included and is 1, *accuracy* can be used to specify the desired fractional accuracy. See Details about algorithms.

If *x* is complex, a complex result is returned. In this case, *algorithm* and *accuracy* are ignored. The order *n* can be fractional, and must be real.

Details

The *algorithm* parameter has three options, each selecting a different calculation method:

Algorithm	What You Get
0 (default)	Uses a calculation method that has fractional accuracy better than 10^{-6} everywhere and is generally better than 10^{-8}. This method does not handle fractional order *n*; the order is truncated to an integer before the calculation is performed. Algorithm 0 is fastest by a large margin.
1	Allows fractional order. The calculation is performed using methods described in *Numerical Recipes in C*, 2nd edition, pp. 240-245. Using algorithm 1, *accuracy* specifies the fractional accuracy that you desire. That is, if you set *accuracy* to 1e-7 (that is, 10^{-7}), that means that you wish that the absolute value of (f_{actual} - $f_{returned}$)/f_{actual} be better than 10^{-7}. Asking for less accuracy gives some increase in speed.