Algorithm	What You Get
	You pay a heavy price for higher accuracy or fractional order. When *algorithm* is nonzero, calculation time is increased by an order of magnitude for small *x*; at larger *x* the penalty is even greater. If accuracy is greater than 10^{-8} and *n* is an integer, algorithm 0 is used. The algorithm calculates bessI and bessK simultaneously. Both values are stored, and if a call to bessI is followed by a call to bessK (or bessK is followed by bessI) with the same *n*, *x*, and *accuracy* the previously-stored value is returned, making the second call very fast.
2	Fractional order is allowed. The calculation is performed using code from the SLATEC library. The accuracy achievable is often better than algorithm 1, but not always. Algorithm 2 is 1.5 to 3 times faster than algorithm 1, but still slower than algorithm 0. The accuracy parameter is ignored.

The achievable accuracy of algorithms 1 and 2 is a complicated function *n* and *x*. To see a summary of achievable accuracies choose File→Example Experiments→Testing and Misc→Bessel Accuracy menu item.

bessJ

bessJ(*n*, *x* [, *algorithm* [, *accuracy*]])

Obsolete — use **Besselj**.

The bessJ function returns the Bessel function of the first kind, J*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

See the **bessI** function for details on algorithms, accuracy and speed of execution.

When *algorithm* is 1, pairs of values for bessJ and bessY are calculated simultaneously. The values are stored, and a subsequent call to bessY after a call to bessJ (or vice versa) with the same *n*, *x*, and *accuracy* will be very fast.

bessK

bessK(*n*, *x* [, *algorithm* [, *accuracy*]])

Obsolete — use **Besselk**.

The bessK function returns the modified Bessel function of the second kind, K*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

See the **bessI** function for details on algorithms, accuracy and speed of execution.

When *algorithm* is 1, pairs of values for bessJ and bessY are calculated simultaneously. The values are stored, and a subsequent call to bessY after a call to bessJ (or vice versa) with the same *n*, *x*, and *accuracy* will be very fast.