

hyperG2F1

See Also

The **hyperG0F1**, **hyperG2F1**, and **hyperGPFQ** functions.

References

The PFQ algorithm was developed by Warren F. Perger, Atul Bhalla, and Mark Nardin.

hyperG2F1

hyperG2F1(a, b, c, z)

The hyperG2F1 function returns the confluent hypergeometric function

$${}_2F_1(a, b, c, z) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n z^n}{(c)_n n!}$$

$${}_2F_1(a, b, c; z) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n z^n}{(c)_n n!},$$

where $(a)_n$ is the Pochhammer symbol

$$(a)_n = a(a+1)\dots(a+n-1).$$

Note: The series evaluation may be computationally intensive. You can abort the computation by pressing the **User Abort Key Combinations**.

See Also

The **hyperG0F1**, **hyperG1F1**, and **hyperGPFQ** functions.

References

The PFQ algorithm was developed by Warren F. Perger, Atul Bhalla, and Mark Nardin.

hyperGNoise

hyperGNoise(m, n, k)

The hyperGNoise function returns a pseudo-random value from the hypergeometric distribution whose probability distribution function is

$$f(x, m, n, k) = \frac{\binom{n}{x} \binom{m-n}{k-x}}{\binom{m}{k}}$$

where m is the total number of items, n is the number of marked items, and k is the number of items in a sample.

The random number generator initializes using the system clock when Igor Pro starts. This almost guarantees that you will never repeat a sequence. For repeatable "random" numbers, use **SetRandomSeed**. The algorithm uses the Mersenne Twister random number generator.

See Also

SetRandomSeed, **StatsHyperGCDF**, and **StatsHyperGPDF**.

Chapter III-12, **Statistics** for a function and operation overview.

Noise Functions on page III-390.

hyperGPFQ

hyperGPFQ(waveA, waveB, z)

The hyperGPFQ function returns the generalized hypergeometric function