$$ {}_pF_q\left(\left\{a_1,\ldots a_p\right\},\left\{b_1,\ldots b_q\right\};z\right)=\sum_{n=0}^{\infty}\frac{(a_1)_n(a_2)_n\ldots(a_p)_n z^n}{(b_1)_n(b_2)_n\ldots(b_q)_n n!}, $$

where $(a)_n$ is the Pochhammer symbol

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

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

See Also

hyperG0F1, **hyperG1F1**, **hyperG2F1**

CosIntegral, **ExpIntegralE1**, **SinIntegral**

References

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

i

```
i
```

The i function returns the loop index of the inner most iterate loop in a macro. Not to be used in a function. iterate loops are archaic and should not be used.

ICA

ICA [*flags*] *srcWave*

The ICA operation performs independent component analysis using the FastICA algorithm. Input data is in the form of a 2D wave where each column represents the equivalent of a single data acquisition channel. The results of the operation are stored in the waves M_ICAComponents, M_ICAUnMix and M_matrixW in the current data folder.

The ICA operation was added in Igor Pro 7.00.

Flags

/A=*alpha*	*alpha* is a constant used as a factor in the argument of the logCosh function. It is not used with the exp function. *alpha* is in the range [1,2] and its default value is 1. You will rarely need to change this value to affect the rate of convergence or the quality of the results.
/CF=num	Specifies the contrast function, also called the non-quadratic function, used by ICA. *num*=0: logCosh (default) *num*=1: exp
/COLS	Preconditions the input by subtracting the mean and then normalizing the input on a column-by-column basis. The algorithm appears to converge and produce better results when this flag is used. As of Igor Pro 9.00, the input is preconditioned by default so /COLS is no longer required or recommended.
/DFLT	Use the deflation/vector method where iterations solve for a single vector of the "unmixing" matrix at a time. By default the operation uses the matrix method which solves for the complete unmixing matrix at one time.

/PCA	Save the output of the "PCA" stage which is also the form of the data before the fastICA iterations. The SVD U matrix is saved in the wave M_PCA and the eigevalues are saved in the wave W_PCAEV. Both are created in the current data folder.
/Q	Quiet mode; do not print anything in the history.
/TOL=*tolerance*	The tolerance value is used to determine when iterations converge. In the deflation/vector method the tolerance measures the difference between the values of vectors in consecutive iterations. In the matrix method the tolerance measures the average deviation of all components. By default tolerance = 1e-5 for both methods.
/WINT=*w*	Provides an initial unmixing matrix W. If you do not provide this matrix the algorithm initializes using enoise. The wave *w* must be 2D having the same number type as *srcWave* and having dimensions nCols x nCols, where nCols is the number of columns of *srcWave*. Providing an initial matrix is useful if you have obtained one from a previous set of iterations which may have converged using inadequate tolerance.
/Z	No error reporting.

Details

srcWave is a 2D wave of nRows by nCols. It must be a single or double precision real-valued wave containing no NaNs or INFs. Each column of *srcWave* corresponds to a single data acquisition channel that is assumed to consist of a linear superposition of independent components. This can be expressed as a matrix product

X=A (S^t)

where **S** is an nRows by nCols matrix of independent components, **^t** denotes the transpose, **A** is an nCols by nCols mixing matrix and **X** is the "mixed" input. The ICA operation attempts to find the independent components of **S** from the transformation

S=W X

so that the mutual information between the resulting columns of **S** is minimized. Since mutual information is not affected by a multiplication of components by scalar constants, the resulting independent components can be specified up to a scalar factor.

The operation uses the FastICA algorithm to compute the independent components.

The algorithm has two available methods for computation. The default is to attempt to evaluate the full **W** matrix at once. The second method (/DFLT flag) also known as "deflation" computes one row of **W** at a time. The deflation method might have advantages in cases where there are fewer independent components than there are columns in the input.

Example

```
// Create the source
Make/O/N=(1000,3) ddd
ddd[][0]=sin(2*pi*x/13)
ddd[][1]=sin(2*pi*x/17)
ddd[][2]=sin(2*pi*x/23)

// Create mixing matrix
Make/O/N=(3,3) AA
AA[0][0]= {0.291,0.6557,-0.5439}
AA[0][1]= {0.5572,0.3,-0.2}
AA[0][2]= {-0.1,-0.7,0.4}

// Do the mixing
MatrixOp/O xx=ddd x AA

// Try the ICA
ICA/DFLT/COLS xx
Display M_ICAComponents[][0]
Display M_ICAComponents[][1]
Display M_ICAComponents[][2]
```