The algorithm iterates until either the tolerance is met or the number of iterations, specified by /ITER or the default value of 1E7, is exceeded. The algorithm also terminates if the the computed norm remains relatively constant:

```
(prevIterationNorm-curIterationNorm)/prevIterationNorm < 1e-16.
```

Output Variables

MatrixFactor sets these automatically created variables:

V_flag	Set to 0 if the operation succeeded or to a non-zero error code.
V_avg	The last average Frobenius norm (Frobenius sum divided by the number of points in *srcWave*).
V_iterations	The number of iterations executed before the algorithm terminated.

Example

```
Function DemoMatrixFactor()
   Make/O/N=(10,8)/D matA0 = 1+abs(enoise(3))
   Make/O/N=(8,12)/D matB0 = 1+abs(enoise(3))
   MatrixOP/O matX0=matA0 x matB0
   MatrixFactor/COMC=8/DSTA = biMatA/DSTB=biMatB matX0
   MatrixOP/O/P=1 aa = sum(sq(matX0- biMatA x biMatB))/120     // Check norm
End
```

References

Nikulin, V., Tian-Hsiang Huang, S. Ng, S. Rathnayake and G. J. McLachlan. "A Very Fast Algorithm for Matrix Factorization." *ArXiv* abs/1011.0506 (2010).

MatrixFilter

MatrixFilter [*flags*] ***Method dataMatrix***

The MatrixFilter operation performs one of several standard image filter type operations on the destination *dataMatrix*.

Note: The parameters below are also available in ImageFilter. See **ImageFilter** for additional parameters.

Parameters

Method selects the filter type. *Method* is one of the following names:

avg	*nxn* average filter.
FindEdges	3x3 edge finding filter.
gauss	*nxn* gaussian filter.
gradN, gradNW, gradW, gradSW, gradS, gradSE, gradE, gradNE	3x3 North, NorthWest, West, … pointing gradient filter.
median	*nxn* median filter. You can assign values other than the median by specifying the desired rank using the /M flag.
min	*nxn* minimum rank filter.
max	*nxn* maximum rank filter.
NanZapMedian	*nxn* filter that only affects data points that are NaN. Replaces them with the median of the *nxn* surrounding points. Unless /P is used, automatically cycles through matrix until all NaNs are gone or until *cols***rows* iterations.
point	3x3 point finding filter `8*center-outer`.
sharpen	3x3 sharpening `filter=(12*center-outer)/4`.