stdDevWave is a wave containing the standard deviation of the noise of the Y data on a point-by-point basis. It is used for the smoothing spline only. *stdDevWave* must have the same number of points as the Y data wave.

If neither /S nor /SWAV are present, Interpolate2 arbitrarily assumes an *s* equal to .05 times the amplitude of the Y data.

/T=*t* Controls the type of interpolation performed.

t=1: Linear interpolation

t=2: Cubic spline interpolation (default)

t=3: Smoothing spline interpolation

/X=*xDest* Specifies the X destination wave.

If /X is present the output is an XY pair.

If /X is omitted the output is a waveform.

The X destination wave may or may not exist when Interpolate2 is called except for "X from dest" mode (/I=3) when it must exist. Interpolate2 overwrites it if it exists.

/Y=*yDest* Specifies the Y destination wave name.

If you omit /Y, a default wave name is generated. The name of the default wave is the name of the source Y wave plus "_L" for linear interpolation, "_CS" for cubic spline or "_SS" for smoothing spline.

The Y destination wave may or may not exist when Interpolate2 is called. Interpolate2 overwrites it if it exists.

See Also

The Interpolate2 Operation on page III-115

Demos

Choose Files→Example Experiments→Feature Demos→Interpolate2 Log Demo.

References

"*Numerical Recipes in C*" for cubic spline.

"Smoothing by Spline Functions", Christian H. Reinsch, *Numerische Mathematic* 10, 177-183 (1967).

Interpolate3D

```
Interpolate3D [/Z ] /RNGX={x0,dx,nx}/RNGY={y0,dy,ny}/RNGZ={z0,dz,nz}
   /DEST=dataFolderAndName, triangulationWave=tWave, srcWave=sWave
```

The Interpolate3D operation uses a precomputed triangulation of *sWave* (see **Triangulate3D**) to calculate regularly spaced interpolated values from an irregularly spaced source. The interpolated values are calculated for a lattice defined by the range flags /RNGX, /RNGY, and /RNGZ. *sWave* is a 4 column wave where the first three columns contain the spatial coordinates and the fourth column contains the associated scalar value. Interpolate3D is essentially equivalent to calling the **Interp3D** function for each interpolated point in the range but it is much more efficient.

Parameters

triangulationWave=*tWave*

Specifies a 2D index wave, *tWave*, in which each row corresponds to one tetrahedron and each column (tetrahedron vertex) is represented by an index of a row in *sWave*. Use **Triangulate3D** with /OUT=1 to obtain *tWave*.

srcWave=*sWave* Specifies a real-valued 4 column 2D source wave, *sWave*, in which columns correspond to *x*, *y*, *z*, f(*x*, *y*, *z*). Requires that the domain occupied by the set of {*x*, *y*, *z*} be convex.

Flags

/DEST=*dataFolderAndName*

Saves the result in the specified destination wave. The destination wave will be created or overwritten if it already exists. *dataFolderAndName* can include a full or partial path with the wave name.

/RNGX={*x0*,*dx*,*nx*} Specifies the range along the X-axis. The interpolated values start at *x0*. There are *nx* equally spaced interpolated values where the last value is at *x0*+(*nx*-1)*dx*. If you would like to interpolate the data for a single plane you can set the appropriate number of values to 1. For example, a YZ plane would have *nx*=1.

/RNGY={*y0*,*dy*,*ny*} Specifies the range along the Y-axis. The interpolated values start at *y0*. There are *nx* equally spaced interpolated values where the last value is at *y0*+(*ny*-1)*dy*. If you would like to interpolate the data for a single plane you can set the appropriate number of values to 1. For example, a XZ plane would have *ny*=1.

/RNGZ={*z0*,*dz*,*nz*} Specifies the range along the Z-axis. The interpolated values start at *z0*. There are *nz* equally spaced interpolated values where the last value is at *z0*+(*nz*-1)*dz*. If you would like to interpolate the data for a single plane you can set the appropriate number of values to 1. For example, a XY plane would have *nz*=1.

/Z No error reporting.

Details

The triangulation wave defines a set of tetrahedra that spans the convex source domain. If the requested range consists of points outside the domain, the interpolated values will be set to NaN. The interpolation process for points inside the convex domain consists of first finding the tetrahedron in which the point resides and then linearly interpolating the scalar value using the barycentric coordinate of the interpolated point.

In some cases the interpolation may result in NaN values for points that are clearly inside the convex domain. This may happen when the preceding Triangulate3D results in tetrahedra that are too thin. You can try using Triangulate3D with the flag /OUT=4 to get more specific information about the triangulation. Alternatively you can introduce a slight random perturbation to the input source wave before the triangulation.

Example

```
Function Interpolate3DDemo()
   Make/O/N=(50,4) ddd=gnoise(20)    // First 3 columns store XYZ coordinates
   ddd[][3]=ddd[p][2]                // Fourth column stores a scalar which is set to z
   Triangulate3D ddd                 // Perform the triangulation
   Wave M_3dVertexList
   Interpolate3D /RNGX={-30,1,80}/RNGY={-40,1,80}/RNGZ={-40,1,80}
         /DEST=W_Interp triangulationWave=M_3dVertexList,srcWave=ddd
End
```

See Also

The **Triangulate3D** operation and the **Interp3D** function. **Interpolation** on page III-114.

References

Schneider, P.J., and D. H. Eberly, *Geometric Tools for Computer Graphics*, Morgan Kaufmann, 2003.