InsertPoints

`InsertPoints` `[` ***`/M=dim /V=value`*** `]` ***`beforePoint, numPoints, waveName`*** `[,` ***`waveName`***`]…`

The InsertPoints operation inserts *numPoints* points in front of point *beforePoint* in each *waveName*. The new points have the value zero.

Flags

/M=*dim*	Specifies the dimension into which elements are to be inserted. Values are 0 for rows, 1 for columns, 2 for layers, 3 for chunks. If /M is omitted, InsertPoints inserts in the rows dimension.
/V=*value*	When used with numeric waves, *value* specifies the value for new elements. If you omit /V, new elements are set to zero. The /V flag was added in Igor Pro 8.00.

Details

Trying to insert points into any but the rows of a zero-point wave results in a zero-point wave. You must first make the number of rows nonzero before anything else has an effect.

See Also

Lists of Values on page II-78.

InstantFrequency

`InstantFrequency [flags]` ***`srcWave`*** **`[ (`*****`startX`*****`,`*****`endX`*****`) ]`**

The InstantFrequency operation computes the instanteous frequency, and optionally the instantaneous amplitude, of srcWave. InstantFrequency was added in Igor Pro 9.00.

InstantFrequency creates an output wave whose name and location depends on the /DEST and /OUT flags as described under ***InstantFrequency Output Wave*** below.

Parameters

srcWave specifies the wave to be analyzed.

[*startX*,*endX*] is an optional subrange to analyze in point numbers.

(*startX*,*endX*) is an optional subrange to analyze in X values.

If you omit the subrange, *startX* defaults to the first point in *srcWave* and *endX* defaults to the last point in *srcWave*.

Flags for all Methods

/DEST=*destWave*	Specifies the output wave created by the operation. *destWave* can be a simple wave name, a data folder path plus wave name, or a wave reference to an existing wave. It is an error to specify the same wave as both *srcWave* and *destWave*. If you omit /DEST, the output wave name depends on /OUT. When used in a function, the InstantFrequency operation by default creates a real wave reference for the destination wave. See **Automatic Creation of WAVE References** on page IV-72 for details. See ***InstantFrequency Output Wave*** below for further discussion.
/FREE	Creates a free destination wave (see **Free Waves** on page IV-91). /FREE is allowed only in functions and only if you include /DEST=destWave where destWave is a simple name or an valid wave reference.

/METH=*method* Specifies the method by which the calculation is performed.

method=0:	Osculating Circle (default)
method=1:	Gabor
method=2:	Spectogram using center of mass
method=3:	Spectogram using maximum amplitude

/OUT=*mode* Specifies the type of output to be created.

mode=1:	Frequency only (default)
mode=2:	Amplitude only
mode=3:	Frequency and amplitude
mode=4:	Signed amplitude only
mode=5:	Debugging output

If you omit /DEST, the output wave is created in the current data folder with a name determined by mode as follows:

mode=1:	W_InstantFrequency
mode=2:	W_InstantAmplitude
mode=3:	M_InstantFrequency
mode=4:	W_InstantAmplitude
mode=5:	M_InstantFrequency

Flags for Spectrogram Method (/METH=2) Only

/HOPS=*hopSize* Specifies the offset in points between centers of consecutive source segments. By default *hopSize* is 1 and the transform is computed for segments that are offset by a single points from each other.

/PAD=*newSize* Converts each segment of *srcWave* into a padded array of length *newSize*. The padded array contains the original data at the center of the array with zeros elements on both sides.

/SEGS=*segSize* Sets the length of the segment sampled from *srcWave* in points. The segment is optionally padded to a larger dimension (see /PAD) and multiplied by a window function prior to FFT. *segSize* must be 32 or greater.

Defining *n* as numpnts(*srcWave*), the default segment size, used if you omit /SEGS, is:

n/200 if *n* >= 25600

128 if 130 <= *n* <= 25599

n-2 if *n* <= 129

/WINF=*windowKind* Premultiplies a data segment with the selected window function. The default window is Hanning. See ***Window Functions*** in the documentation for **FFT** for details.

InstantFrequency Output Wave

If you use the /FREE flag then the output wave is created as a free wave using the name or wave reference specified by /DEST=*destWave*.

If you include the /DEST flag and omit /FREE then the output wave location and name is specified by the *destWave* parameter. *destWave* can be a simple wave name, a data folder path plus wave name, or a wave reference to an existing wave.

If you omit the /DEST and /FREE flags then the output wave is created in the current data folder with the default name W_InstantFrequency (/OUT=1), W_InstantAmplitude (/OUT=2 or 4), or M_InstantFrequency (/OUT=3 or 5), depending on the /OUT flag.

Gabor Method (/METH=1)

The Gabor method uses the Hilbert Transform to synthesize an "analytic" signal that is a copy of the source wave, shifted by 90 degrees, but with the constant ("DC") component removed.

A complex "phasor" wave is generated whose real part is the source wave, and the imaginary part is this analytic signal.

The phase at each time point is computed as atan2(imag(phasor[p]),real(phasor[p])). This phase calculation is affected by any constant component of the source wave. The derivative of this phase with respect to time is the instantanteous frequency.

The instantaneous amplitude is the magnitude of the phasor[p].

See the **HilbertTransform** operation example for an equivalent implementation.

The /OUT=5 debugging output for the Gabor method is:

```
destWave[][0]       // Instant frequency
destWave[][1]       // Instant amplitude
destWave[][2]       // Unwrapped phase wave
destWave[][3]       // Trajectory Y wave (Hilbert transform of input wave)
```

Osculating Circle Method (/METH=0)

The primary drawback of the Gabor method is that any sizeable constant level distorts the phase such that it isn't always increasing, which results in negative frequencies being computed. The Osculating Circle Method does not have this problem.

The Osculating Circle method constructs the same analytic phasor as the Gabor method, but computes the phase and derivative differently in a way that eliminates the constant level distortion: at each point the phasor's previous, current, and next complex values (the "trajectory") are fit to a circle in the complex plane. That circle's origin is used to measure both the phase and amplitude at that point, instead of the origin at (0+i0) that the Gabor method uses.

The /OUT=5 debugging output for the Osculating Circle method is:

```
destWave[][0]       // Instant frequency
destWave[][1]       // Signed instant amplitude
destWave[][2]       // Unwrapped phase wave
destWave[][3]       // Trajectory Y wave (Hilbert transform of input wave)
destWave[][4]       // Trajectory dY wave
destWave[][5]       // Trajectory ddY wave
destWave[][6]       // Trajectory dX wave
destWave[][7]       // Trajectory ddYX wave
destWave[][8]       // Origin Y wave
destWave[][9]       // Origin X wave
```

Spectrogram (/METH=2 or 3)

The Spectogram method for determining instant frequency and amplitude is based on measuring the Short-Time Fourier Transform, the 2D time-frequency representation for a 1D array. The scaled magnitude of the transform is known as the "spectrogram" for time series or "sonogram" in the case of sound input. Methods 2 and 3 are comprised of the following steps:

1. Compute the Short-Time Fourier Transform according to the various spectogram parameters. This results in a 2D wave, where the columns of each row comprise the scaled magnitude spectrum at one point in time.

2. For each spectrum determine where the dominant frequency lies, and its magnitude.

For /METH=2, the dominant frequency is found using the centerOfMass function.

For /METH=3, the dominant frequency is found by locating the maximum value.

The /OUT=5 debugging output for the Spectogram method is the 2D spectogram that would have been used as the output of step 1.

References

Wikipedia: https://en.wikipedia.org/wiki/Instantaneous_phase_and_frequency

Wikipedia: https://en.wikipedia.org/wiki/Gabor_transform