Mismatched Waves

For most applications you will not need to mix waves of different lengths. In fact, doing this is more often the result of a mistake than it is intentional. However, if your application requires mixing you will need to know how Igor handles this.

Let's consider the case of assigning the value of one wave to another with a command such as

```
wave1 = wave2
```

In this assignment, there is no explicit indexing, so Igor evaluates the expression as if you had written:

```
wave1 = wave2[p]
```

If wave2 has more points than wave1, the extra points have no effect on the assignment since p ranges from 0 to n-1, where n is the number of points in wave1.

If wave2 has fewer points than wave1 then Igor will try to evaluate wave2[p] for values of p greater than the length of wave2. In this case, it returns the value of the last point in wave2 during the wave assignment statement but also raises an "index out of range" error. Previous versions of Igor did not raise this error.

It may be that you actually want the values in wave1 to span the values in wave2 by interpolating between values in wave2. To get Igor to do this, you must explicitly index the appropriate X values on the right side. For instance, if you have two waves of different lengths, you can do this:

```
big = small[p*(numpnts(small)-1)/(numpnts(big)-1)]
```

NaNs, INFs and Missing Values

The data value of a point in a floating point numeric wave is normally a finite number but can also be a NaN or an INF. NaN means "not a number". An expression returns the value NaN when it makes no sense mathematically. For example, `log(-1)` returns the value NaN. You can also set a point to NaN, using a table or a wave assignment statement, to represent a missing value. An expression returns the value INF when it makes sense mathematically but has no finite value. `log(0)` returns the value -INF.

The IEEE floating point standard defines the representation and behavior of NaN values. There is no way to represent a NaN in an integer wave. If you attempt to store NaN in an integer wave, you will store a garbage value.

Comparison operators do not work with NaN parameters because, by definition, NaN compared to anything, even another NaN, is false. Use **numtype** to test if a value is NaN.

Igor ignores NaNs and INFs in curve fit and wave statistics operations. NaNs and INFs have no effect on the scaling of a graph. When plotting, Igor handles NaNs and INFs properly, as missing and infinite values respectively.

Igor does *not* ignore NaNs and INFs in many other operations, especially those that are DSP related such as FFT. In general, any operation that numerically combines all or most of the data points from a wave will give meaningless results if one or more points is a NaN or INF. Notable examples include the **area** and **mean** functions and the **Integrate** and **FFT** operations. Some operations that only mix a few points such as Smooth and Differentiate will "contaminate" only those points in the vicinity of the NaN or INF. You can use the Interpolate operation (Analysis menu) to create a NaN-free version of a wave.

If you get NaNs from functions such as **area** or **mean** or operations such as **Convolve** or any other functions or operations that sum points in waves, it indicates that some of the points in the wave are NaN. If you get NaNs from curve-fitting results, it indicates that Igor's curve fitting has failed. See **Curve Fitting Troubleshooting** on page III-266 for troubleshooting tips.

See **Dealing with Missing Values** on page III-112 for techniques for dealing with NaNs.