the destination may be an XY pair in which case the interpolation is done at the data values stored in the X destination wave.

Here is an example using a waveform as the destination:

```
Make /O /N=20 wave0                        // Generate source data
SetScale x 0, 2*PI, wave0
wave0 = sin(x) + gnoise(.1)
Display wave0
ModifyGraph mode=3
Make /O /N=1000 dest0                      // Generate dest waveform
SetScale x 0, 2*PI, dest0
AppendToGraph dest0
ModifyGraph rgb(dest0)=(0,0,65535)
Interpolate2 /T=2 /I=3 /Y=dest0 wave0      // Do cubic spline interpolation
```

If your destination is an XY pair, Interpolate2 creates a sorted version of these values and then calculates its output at the X destination wave's values. If the X destination wave was originally reverse-sorted, as determined by examining its first and last values, Interpolate2 reverses the output after doing the interpolation to restore the original ordering.

In X From Dest mode with an XY pair as the destination, the X wave may contain NaNs. In this case, Interpolate2 creates an internal copy of the XY pair, removes the points where the X destination is NaN, does the interpolation, and copies the results to the destination XY pair. During this last step, Interpolate2 restores the NaNs to their original locations in the X destination wave. Here is an example of an XY destination with a NaN in the X destination wave:

```
Make/O xData={1,2,3,4,5}, yData={1,2,3,4,5}           // Generate source data
Display yData vs xData
Make/O xDest={1,2,NaN,4,5}, yDest={0,0,0,0,0}         // Generate dest XY pair
ModifyGraph mode=3,marker=19
AppendToGraph yDest vs xDest
ModifyGraph rgb(yDest)=(0,0,65535)
Interpolate2 /T=1 /I=3 /Y=yDest /X=xDest xData, yData // Do interpolation
```

Differentiation and Integration

The **Differentiate** operation (see page V-158) and **Integrate** operation (see page V-446) provide a number of algorithms for operation on one-dimensional waveform and XY data. These operations can either replace the original data or create a new wave with the results. The easiest way to use these operations is via dialogs available from the Analysis menu.

For most applications, trapezoidal integration and central differences differentiation are appropriate methods. However, when operating on XY data, the different algorithms have different requirements for the number of points in the X wave. If your X wave does not show up in the dialog, try choosing a different algorithm or click the help button to see what the requirements are.

When operating on waveform data, X scaling is taken into account; you can turn this off using the /P flag. You can use the **SetScale** operation (see page V-853) to define the X scaling of your Y data wave.

Although these operations work along just one dimension, they can be targeted to operate along rows or columns of a matrix (or even higher dimensions) using the /DIM flag.

The Integrate operation replaces or creates a wave with the numerical integral. For finding the area under a curve, see **Areas and Means** on page III-120.

Areas and Means

You can compute the area and mean value of a wave in several ways using Igor.

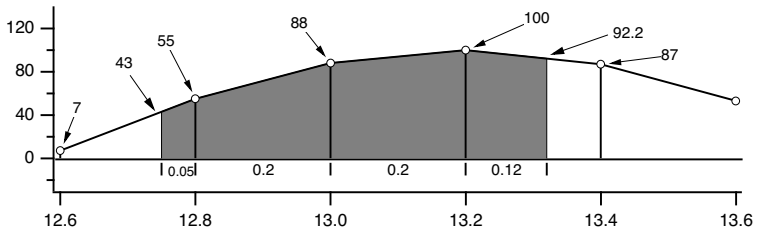
Perhaps the simplest way to compute a mean value is with the Wave Stats dialog in the Analysis menu. The dialog is pictured under **Wave Statistics** on page III-122. You select the wave, type in the X range (or use the current cursor positions), click Do It, and Igor prints several statistical results to the history area. Among them is V_avg, which is the average, or mean value. This is the same value that is returned by the **mean** function (see page V-586), which is faster because it doesn't compute any other statistics. The mean function returns NaN if any data within the specified range is NaN. The **WaveStats** operation, on the other hand, ignores such missing values.

WaveStats and the **mean** function use the same method for computing the mean: find the waveform values within the given X range, sum them together, and divide by the number of values. The X range serves only to select the values to combine. The range is rounded to the nearest point numbers.

If you consider your data to describe discrete values, such as a count of events, then you should use either WaveStats or the mean function to compute the average value. You can most easily compute the total number of events, which is an area of sorts, using the **sum** function (see page V-1007). It can also be done easily by multiplying the WaveStats outputs V_avg and V_npnts.

If your data is a sampled representation of a continuous process such as a sampled audio signal, you should use the **faverage** function to compute the mean, and the area function to compute the area. These two functions use the same linear interpolation scheme as does trapezoidal integration to estimate the waveform values between data points. The X range is *not* rounded to the nearest point. Partial X intervals are included in the calculation through this linear interpolation.

The diagram below shows the calculations each function performs for the data shown. The two values 43 and 92.2 are linear interpolation estimates.



Comparison of area, faverage and mean functions over interval (12.75,13.32)

`WaveStats/R=(12.75,13.32) wave`		
`V_avg`	= (55+88+100+87)/4 = 82.5	
`mean(wave,12.75,13.32)`	= (55+88+100+87)/4 = 82.5	
`area(wave,12.75,13.32)`	= 0.05 · (43+55) / 2	first trapezoid
	+ 0.20 · (55+88) / 2	second trapezoid
	+ 0.20 · (88+100) / 2	third trapezoid
	+ 0.12 · (100+92.2) / 2	fourth trapezoid
	= 47.082	
`faverage(wave,12.75,13.32)`	= area(wave,12.75,13.32) / (13.32-12.75)	
	= 47.082/0.57 = 82.6	

Note that only the area function is affected by the X scaling of the wave. faverage eliminates the effect of X scaling by dividing the area by the same X range that area multiplied by.