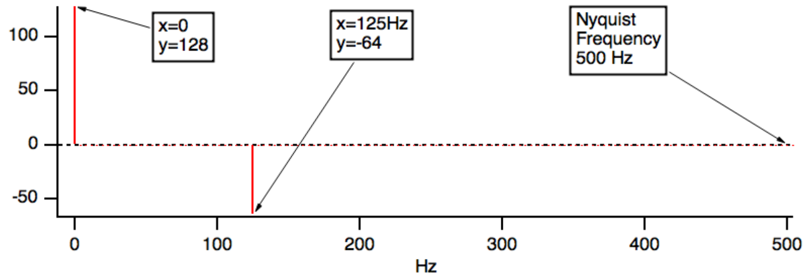

The Fourier Transform would predict a zero-frequency ("DC") result of 1, which is what we get when we divide the FFT value of 128 by the number of input values which is also 128. In general, the Fourier Transform value at zero frequency is:

$$\textit{Fourier Transform Amplitude}(0) = \frac{1}{N} \cdot real(r2polar(wave_{FFT}(0)))$$

The Fourier Transform would predict a spectral peak at -125Hz of amplitude (-0.5 + i0), and an identical peak in the positive spectrum at +125Hz. The sum of those expected peaks would be (-1+0·i).

(This example is contrived to keep the imaginary part 0; the real part is negative because the input signal contains -cos(…) instead of + cos(…).)

Igor computed only the positive spectrum peak, so we double it to account for the negative frequency peak twin. Dividing the doubled peak of -128 by the number of input values results in (-1+i0), which agrees with the Fourier Transform prediction. In general, the Fourier Transform value at a nonzero frequency f is:

$$\textit{Fourier Transform Amplitude}(f) = \frac{2}{N} \cdot real(r2polar(wave_{FFT}(f)))$$

The only exception to this is the Nyquist frequency value (the last value in the one-sided FFT result), whose value in the one-sided transform is the same as in the two-sided transform (because, unlike all the other frequency values, the two-sided transform computes only one Nyquist frequency value). Therefore:

$$\textit{Fourier Transform Amplitude}(f_{Nyquist}) = \frac{1}{N} \cdot real(r2polar(wave_{FFT}(f_{Nyquist})))$$

The frequency resolution $dX_{FFT} = 1/(N_{original} \cdot dx_{original})$, or 1/(128*1e-3) = 7.8125 Hz. This can be verified by executing:

```
Print deltax(wave0)
```

Which prints into the history area:

```
  7.8125
```

You should be aware that if the input signal *is not* a multiple of the frequency resolution (our example *was* a multiple of 7.8125 Hz) that the energy in the signal will be divided among the two closest frequencies in the FFT result; this is different behavior than the continuous Fourier Transform exhibits.

Phase Polarity

There are two different definitions of the Fourier transform regarding the phase of the result. Igor uses a method that differs in sign from many other references. This is mainly of interest if you are comparing the result of an FFT in Igor to an FFT in another program. You can convert from one method to the other as follows:

```
FFT wave0; wave0=conj(wave0)  // negate the phase angle by changing
                              // the sign of the imaginary component.
```

Effect of FFT and IFFT on Graphs

Igor displays complex waves in Lines between points mode by default. But, as demonstrated above, if you perform an FFT on a wave that is displayed in a graph and the display mode for that wave is lines between