format. The addition of an imaginary *radix point* after the MSB forces the numerical representation into, what is termed *signed-fractional* format. Signed-integer and signed-fractional formats are the representations most often found inside digital signal processing (DSP) chips; especially signed-fractional. Most fixed-point, DSP operations are optimised for this latter representation.

The Discrete Fourier Transform

The Discrete Fourier Transform was mentioned in an earlier chapter. However, braced as we now are with a better understanding of digital signals, it is time to revisit this important technique. The Fourier Transform exists because an electrical signal may be described just as accurately in two different ways; in the *frequency base* (or *frequency domain*) and in the *time base* (or *time domain*). The Fourier Transform is the analysis tool we use to get from the time domain description to a frequency domain description.

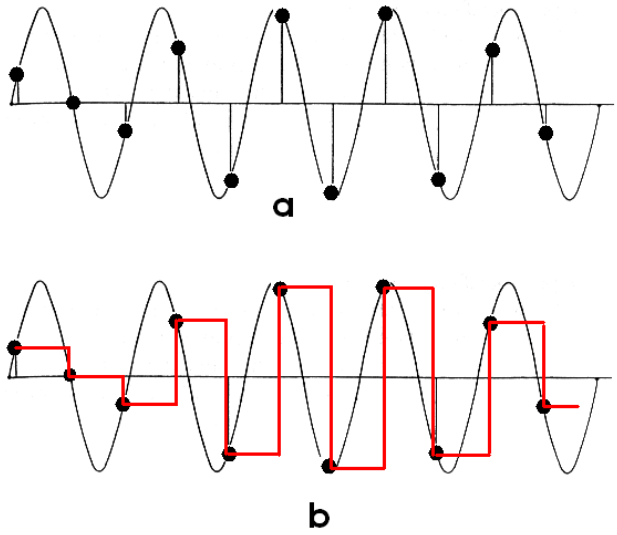

Figure 7 - In the digital domain, signals are not continuous. Instead they are a stream of instantaneous readings of voltage at a vanishingly small instant of time. Never be tempted to think of digital samples as a staircase waveform, or be tempted to draw lines between the dots which represent the samples as illustrated in (b)

In the digital domain, signals are not continuous as in an analogue circuit, they are said to be *sampled*. As we have seen, signals (from, for example, a microphone) are “snapshots”. That’s to say, a stream of contiguous, instantaneous readings of voltage at a vanishingly small instant of time (Figure 7a). Never be tempted to think of digital samples as a staircase waveform, or be tempted to draw lines between the dots which represent the samples as illustrated in Figure 7(b). Despite the fact that you will often see this in explanations of digital signals, it is a mistake. That is because there is *nothing between the samples of a time-sampled signal*. It’s not that we *don’t know* what's between the sampling points: there *is nothing* between the sampling instants. This may seem rather a philosophical point, but it’s profoundly important. In the sampled world of digital signals, guitar strings snap from one position to another and drum sticks descend onto the drum membrane in a series of dislocated movements, each position frozen in time like a movie film.

There’s an important corollary here that comes about because of this apparently rather arcane notion about digital signals. It is this: the *discontinuous time-base* in which digital signals exist, results in a *discontinuous frequency base* when these signals are transformed to the frequency domain. And that means that, although the Discrete Fourier Transform is still a cumbersome tool, it is vastly simplified compared with its continuous cousin because, the discrete form of the Fourier Transform can be used to transform a series of

discrete, sampled, amplitude points in the time domain into a series of *discrete frequencies* in the frequency domain. It is written like this:

$$\mathbf{X}(m) = \sum_{n=0}^{N-1} x(n)\, \mathbf{e}^{(-j2\pi nm/N)}$$

Once again, we can use Euler's formula to express the equation like this:

$$\mathbf{X}(m) = \sum_{n=0}^{N-1} x(n)\, [\cos(2\pi nm/N) - j \sin(2\pi nm/N)]$$

where $\mathbf{X}(m)$ = the m^{th} DFT output component

m = the *index* of the DFT output in the frequency base, m = 0, 1, 2, 3,....., (N-1),

$x(n)$= the sequence of input samples, $x(0), x(1), x(2), x(3),$ etc.,

n = the time base index of the input samples, n = 0, 1, 2, 3, 4, ... (N-1),

j = √-1 , and N = the number of samples of the input sequence *and* the number of frequency points in the DFT output.

The exact frequencies of the different analysis points in the DFT output depend upon the sampling rate of the time based signal (***fs***) and the number of samples included in the analysis (***N***). The fundamental analysis frequency is given by ***Fs/N*** and the ***N*** frequency points are multiples of this frequency (including a multiple of zero, to give a DC term). So, if our sampling frequency is 44.1kHz as it is on a CD, and we decide to do a Fourier analysis over 16 samples of audio, the precise analysis frequencies will be,

Fs/N = 2.75625 kHz

2.75625 kHz . 0 = 0Hz (DC)
2.75625 kHz . 1 = 2.75625kHz
2.75625 kHz . 2 = 5.5125kHz
2.75625 kHz . 3 = 8.26875kHz
2.75625 kHz . 4 = 11.025 kHz
. . .
. . .
. . .

2.75625 kHz . 15 = 41.34375kHz

Leakage and Windowing

You may be thinking, it's all well and good that the DFT allows for the simplification that the analysis can be determined by a finite number of calculating steps and produce an analysis into a series of discrete frequencies, but real-world signals aren't that convenient. Just because my analysis chooses to represent the first bar of Beethoven's 5th Symphony in sixteen discrete frequencies, doesn't mean that there are only sixteen frequencies in the signal. And you'd be right!

The result of trying to force the real world into the straightjacket of the Discrete Fourier Transform is termed, *leakage*: a signal which falls anywhere but exactly upon the analysis frequencies, will "leak" into all the other frequency analysis points. (In DFT terminology, the frequency analysis points are termed, *frequency bins*.) This leakage has the characteristic that the analysis leaks more into the nearer bins than those further away, as is illustrated in Figure 8. The amount of leakage into neighbouring frequency bins is minimised by various techniques. The first is – evidently – to use a finer analysis comb and increase the number of samples over which the signal is analysed, thereby closer approximating the real frequencies which exist in the original analogue signal. The second technique is termed *windowing*. We already met windowing in relation to the explanation of the continuous transform. In terms of the DFT and the minimisation of

leakage, the idea of the window function is to force a periodicity on the input sequence so that the beginning and end of the analysis window converge gently on a common value.

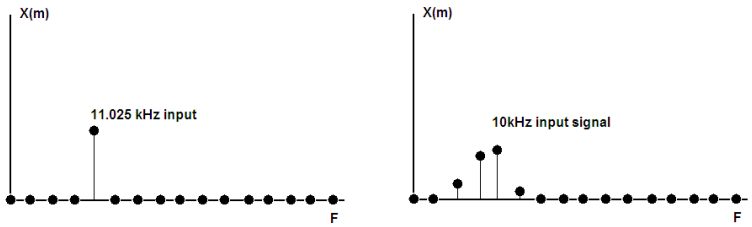

Figure 8 - The phenomenon of *leakage* in the DFT

A practical example

The foregoing may still seem a little arcane. The best way I know to alleviate this feeling, is to experiment with a practical DFT. Not only does this demonstrate the principle, it will introduce a couple of very important issues regarding the DFT. With this book, I have included a simple DFT program as an Excel spreadsheet which is available at www.richardbrice.net. Excel is a wonderful program and has all sorts of uses over and above a financial analysis tool. Modern implementations of spreadsheets, like Excel, include many higher-math functions and very fast graphing tools and are therefore useful in many engineering applications. The alternative is to write programs from scratch, which is involved and demands greater preparation. Figure 9 illustrates the user interface. The program is very simple; it's an eight-point DFT. This is deliberate: so that the process is simple enough to take time to look at the formulae in the cells and see what's going on.

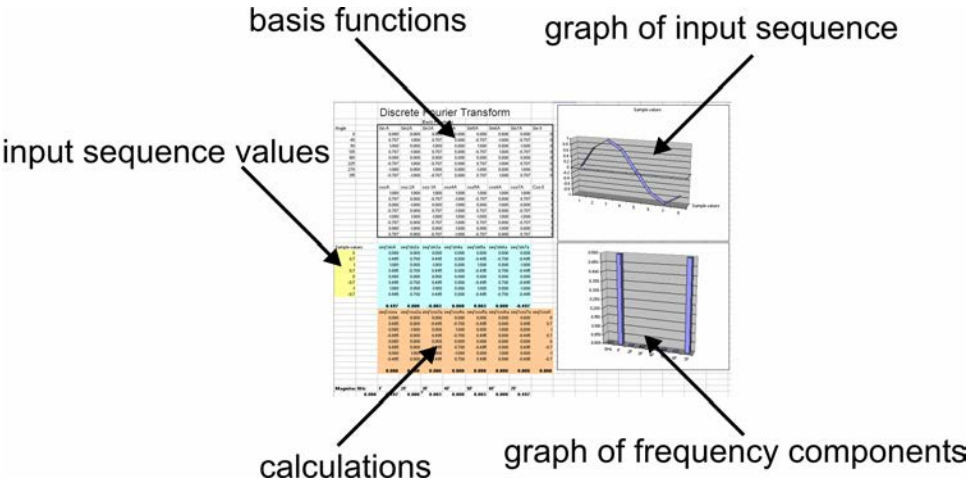

Figure 9 - Simple DFT as an Excel spreadsheet

The 8-value input sequence is entered in the yellow box as illustrated in Figure 9. The sequence is automatically graphed at the top right-hand of the page. (I've broken my own rule and "joined the dots" in the time domain presentation. This is technically wrong but - when an input function only has eight points – it's very difficult to see without doing this.) The DFT is automatically calculated and presented as a bar-chart in the lower right-hand corner of the page. The *basis functions* are shown boxed at the top of the page and the intermediate calculation results are displayed in the blue and salmon coloured areas. In this simple spreadsheet, I have only calculated the *magnitude* of each frequency component; not its phase.

It's illuminating, to try various input-sequences, watch the time-domain display (top right) and see the result immediately appear in the frequency domain (bottom right). I have

illustrated a few examples below. Firstly (Figure 10), an input which is a perfect sine wave at ***F1*** (***F1 = 1/N*** × sample frequency). Note that the analysis illustrates this too, with a bar at ***F1***.

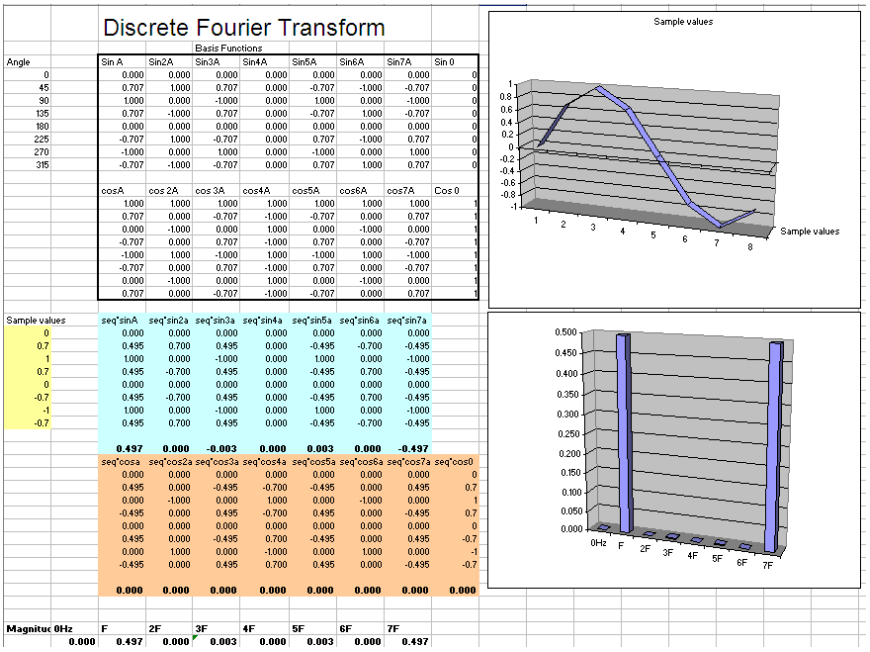

Figure 10 – Sine wave and it DFT

In the second example (Figure 11) the input was calculated to be ***F1 + F2/2***: the analysis confirms this.

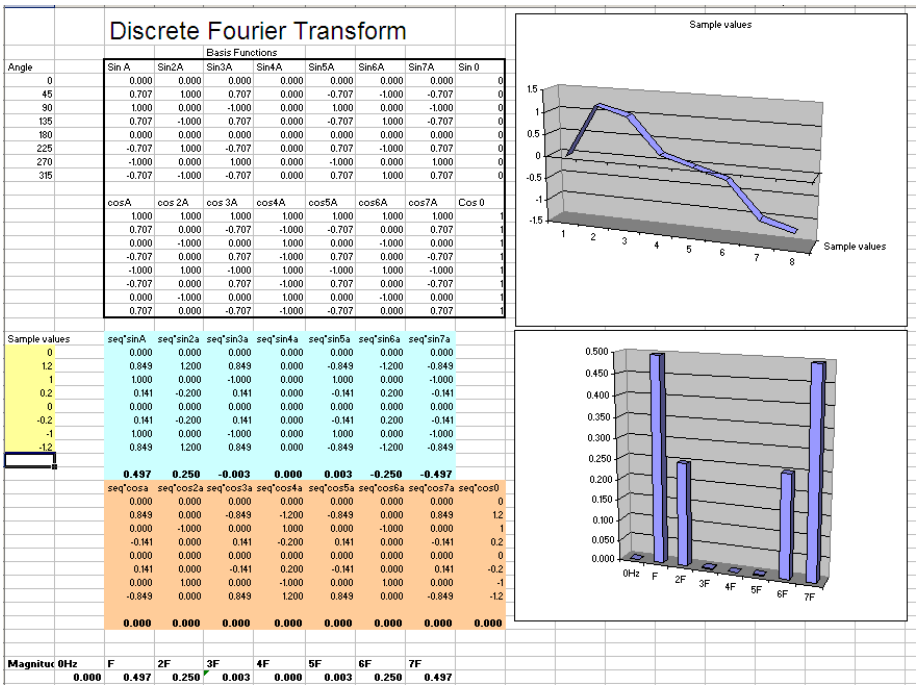

Figure 11 – see text

The third example (Figure 12) has an input sequence which was calculated as the sum of ***F1 + F3***: the frequency domain representation confirms this.

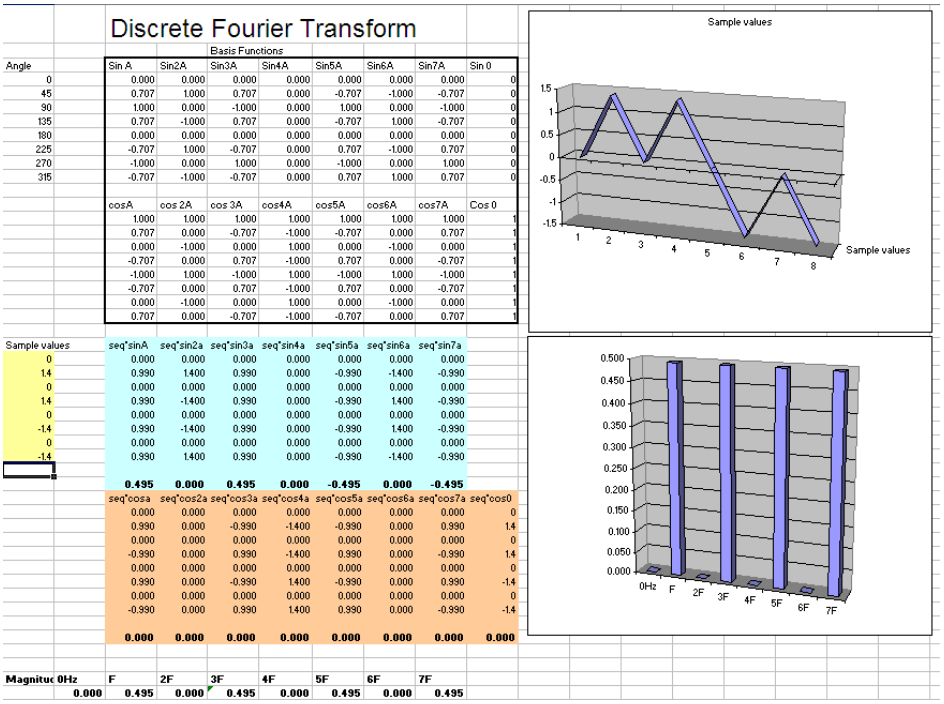

Figure 12 - see text

By now, I hope you're thinking, "What's going on at the top of the frequency domain representations in each case? Sure, when the input was equivalent to the first basis function ***F1***, the result was a bar at ***F1*** (Figure 10), but what about the equally sized bar at 7F?"

We are back to the ambiguity of digital signals. Although we think of the amplitude samples in the time domain as being a continuous function, because they're usually based on the value of some continuous function in the real world (like an audio signal), the digital signal is *discontinuous* and *ambiguous*. When a single frequency in the analogue domain is represented in the sampled domain, its energy exists at the original frequency, *and as sidebands disposed symmetrically about the sampling frequency and its harmonics*. We saw this in Figure 2. Then it was a fanciful idea. But look at Figures 11 and 12. Each of the examples demonstrates that the spectrum of a signal exists (in the sampled domain) both at the original frequencies and as a mirror image, reflected at the sampling frequency (***8F***). Our DFT demonstrates that the phenomenon illustrated in Figure 2 isn't some refined, esoteric idea: the spectrum of a digital signal really does look like this. Its frequency is said to be ambiguous and it is only the action of the reconstruction filter which resolves this ambiguity. Once again, this demonstrates why analogue signals - when represented in the sampled domain - must be limited and never include frequencies higher than half the sampling frequency (sometimes called the *Nyquist frequency limit*). When we want to use a DFT to display the frequency of a continuous input function, only the first half of the analysis in the spreadsheet is performed and displayed. This cuts down the maths by half.

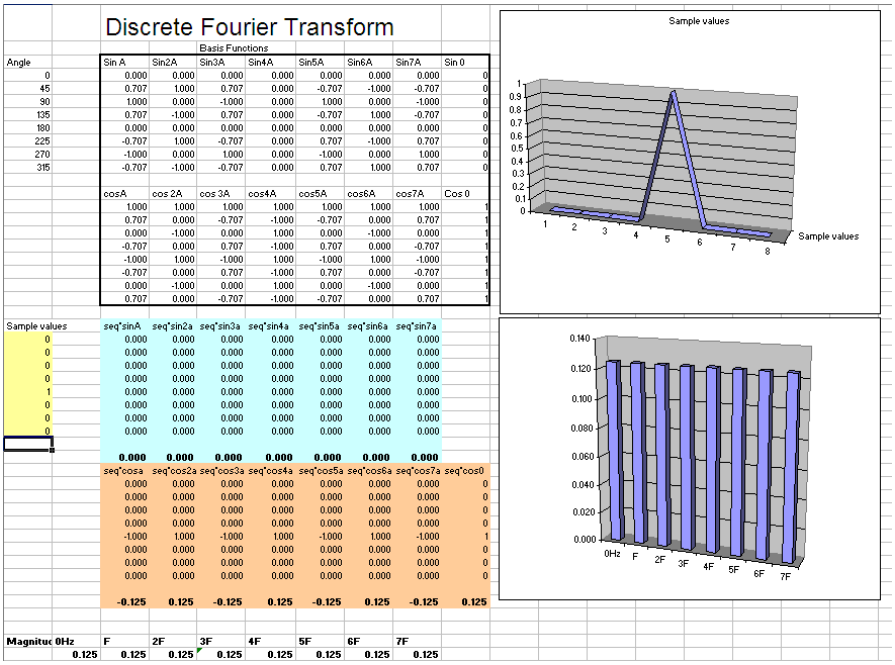

Figure 13 - see text

Two final examples of the DFT spreadsheet are given in Figure 13 and Figure 14. In the first example (Figure 13), the very important input function is **0, 0, 0, 0, 1, 0, 0, 0** is transformed. In words, the transform is derived for a single, very short pulse. Note how this appears in the frequency domain; as a function in which all discrete frequencies are present. We shall come to meet this input function again when we come to look at digital filters.

The last example (Figure 14) demonstrates *leakage*. The input sequence is derived from sine 1.5F. Note how the results leak into the adjacent frequency bins.

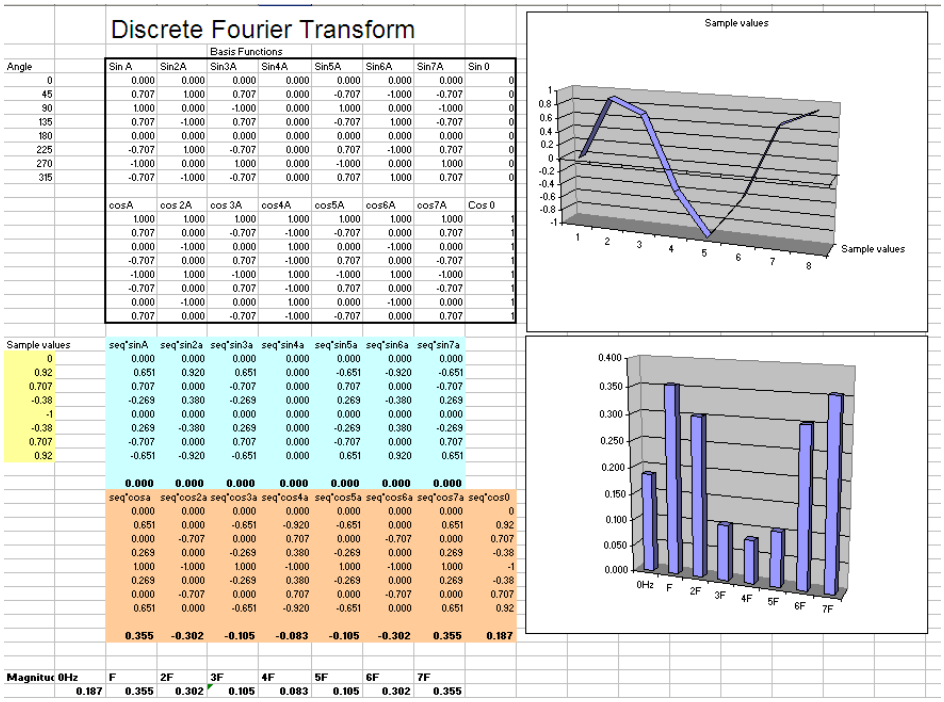

Figure 14 - see text