DSP101 Part 5 Background to Fast Convolution

Introduction

As discussed in Part 4 the ability to do 'fast convolution' depends upon being able to do a Fast Fourier Transform, and of course the inverse transform too. Implementing this involves perhaps unfamiliar notations and concepts, so let's review.

Fourier Analysis

Any analogue (continuous) signal can be analysed and modelled as being made up of sinusoidal harmonics of given amplitudes and phases. There are constraints of course and if you are keen to check on these look up 'Dirichlet conditions'. However, the time domain Fourier series is expressed as a DC component plus many sine and cosine components. Simple sinusoids are symmetrical about the zero amplitude axis and so have a zero average value. If you integrate the Fourier series for any signal you will evaluate the DC component (it's the only component that doesn't have a zero average value). If you multiply both sides of the Fourier series equation by a specific sine or cosine wave, there will be one of two results: either, the specific wave you have introduced will not be present in the original signal and when you integrate the result be still be zero; or, the introduced wave will be the same as one already present in the signal. In this latter case, when you multiply throughout you will get a term involving Sin^2 and this will be all above the zero line and thus have a positive average value. This principle can be used to 'find' the frequency components in a given waveform. This is the 'sieving' action of a Fourier integral. The continuous spectrum of an aperiodic signal is calculated using the Fourier integral effectively performs a Fourier transform. The Fourier transform when applied to a function of time (f(t)) produces the frequency spectrum, a function of frequency (G(ω)). The inverse transform takes you back again.

The exponential Fourier series

Writing down all the sines and cosines that make up a spectrum is a pain! Fortunately there is a notation that makes it 'painless'.

$e^{j\omega t}$= Cos ωt + jSin ωt and $e^{-j\omega}$ = Cos ωt - jSin ωt
You'll recognise this later.

Sampled Data signals

Digitising analogue signals involves an A/D converter and the sample values obtained are often modelled as 'weighted Dirac functions'. A Dirac function is a unit amplitude impulse or spike and a function of time. A sampled sinusoid might be represented as weighted Dirac functions as follows:

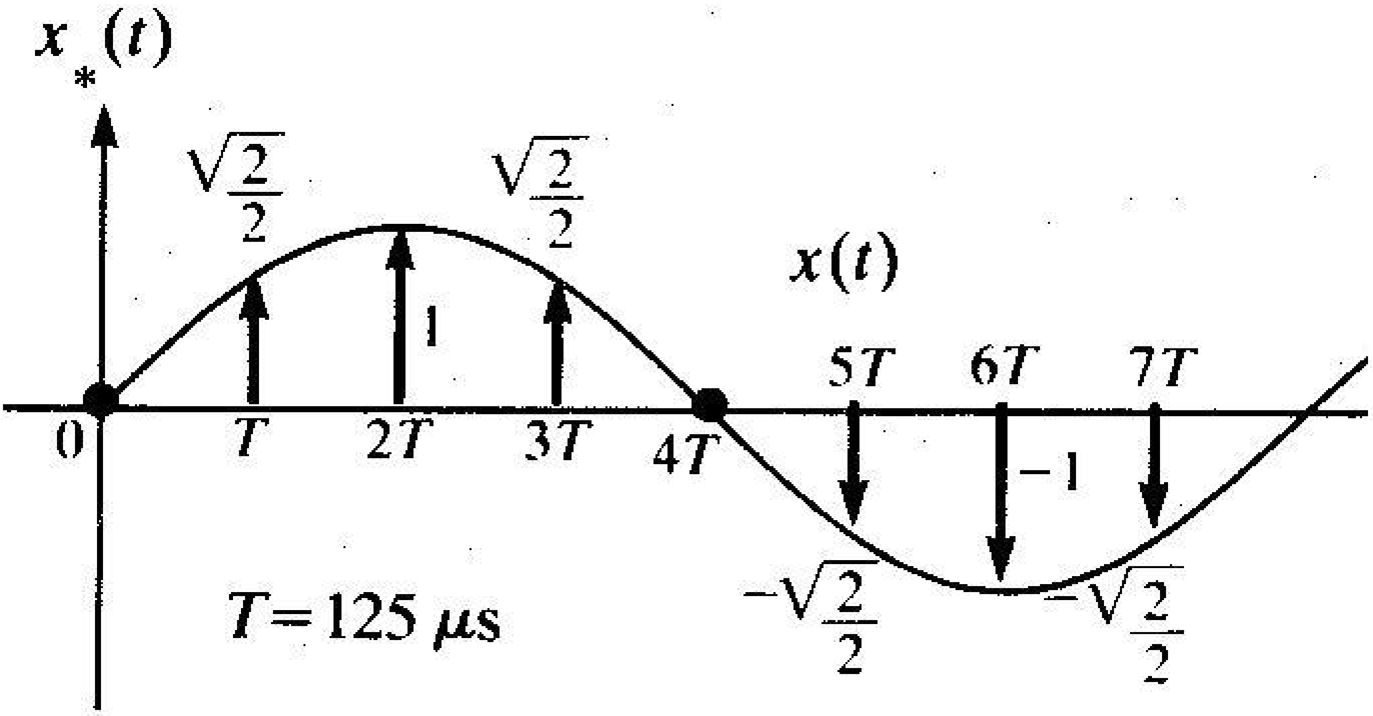

Fig 5.1 Sampled sinusoid (Open University)

The generalised weighted Dirac function representation for a sampled data signal looks like this as a function of time:

f(t)=x0 . δ(t)+ x1 .δ(t-T)+x2 . δ(t-2T)....................

where xn is a sample value, and T= the sampling period, δ is the usual representation of an impulse.

This signal, a series of weighted impulses, will have a frequency spectrum, which can be obtained using the Discrete Fourier Transform (DFT). The DFT is a variation on the Fourier transform, just modified to be appropriate to sampled data e.g. discrete samples.

The DFT and Inverse DFT

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn} \qquad k = 0, \ldots, N$$

X_k are frequency domain values, N=the number of samples in total, n= a particular sample. Σ means summation. So, lets say there are 100 time domain samples (n=0 to 99). There will be 100 (k=0 to 99) output values in the frequency domain. The right hand side of the equation says for input samples 0-99 sum all the samples times the exponential Fourier series (all the sines and cosines etc) etc.

The Inverse transform is:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \qquad n = 0, \dots, N$$

Yes, but how do we actually do it?

Part 6 will talk about the Fast Fourier Transform and how it's generally implemented in terms of calculation order and the normal deconstruction into atomic entities called FFT butterflies.