Reading the Sines:
Sinusoidal Identification and Description using the Short Time Fourier Transform

Jez Wells
Audio Lab, Department of Electronics, University of York, UK

Introduction
Sinusoids are often considered to be the fundamental building blocks of audio signals as well as many other types. Indeed classical (Helmholtz) theory of the ‘sensation of tones’ held that sound quality (or timbre) was entirely determined by the relative levels of summed sinusoids of different frequencies[1]. Research carried out in the 125 years since this work has expanded this definition of timbre to include temporal variation (envelopes) and other sound types such as transients and noise-like components. Although, according to Fourier theory, any signal can be described as a sum of weighted stationary sinusoids it is often desirable to employ a sound model that is more efficient and intuitive. Such a model might very well still seek to identify and describe ‘stable’ sinusoidal components (i.e. those that exist for a relatively long period of time and whose parameters may or may not vary relatively slowly over that period of time) in addition to other types of components such as filtered noise.

This tutorial review outlines some common approaches to identifying such sinusoidal components within the spectrum of an audio signal and presents methods for extracting the parameters, such as frequency, for these components. The intention is to offer a starting point in this area of audio analysis which is accessible to both newcomers to time-frequency analysis and those already experienced in this field. Included in the list of references, cited to credit the originators and developers of the techniques I discuss, are some useful sources for further information about basic concepts discussed and time and frequency domain processing of digital signals in general.

What does a sinusoid look like?
The term **sinusoid** is used to describe a **cosine** or **sine** function of arbitrary phase offset (starting point of oscillation). Two time domain plots are shown below (*x* values are on the horizontal axis, values are on the horizontal axis, *y* values on the vertical). In general for a sinusoidal function at time (t) the signal y(t) is defined by: $y(t) = \sin(\omega t + \phi)$ where ω is the radian frequency and ϕ is the phase offset (in radians)[2].

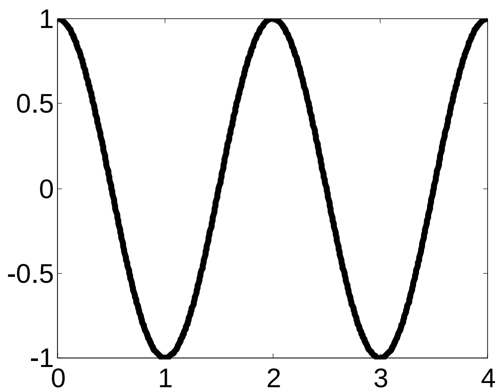


Figure 1: Sine function, $y = \sin(\pi x)$ or $y = \cos\left(\pi x + \frac{3\pi}{2}\right)$

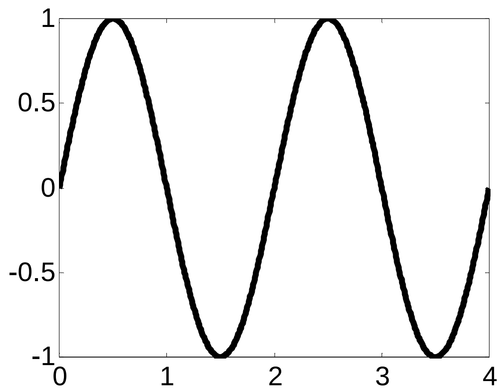
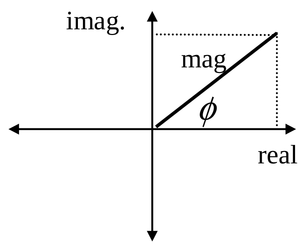


Figure 2: Cosine function, $y = \cos(\pi x)$ or $y = \sin\left(\pi x + \frac{\pi}{2}\right)$

When we perform Fourier analysis with the DFT (discrete Fourier transform) we obtain a frequency representation of our sampled time domain signal in terms of a series of complex numbers. Each complex number represents a different frequency region in the spectrum. The real part of each complex number represents the cosine part of the signal at a particular frequency and the imaginary part represents the sine part of the signal at the same frequency. For figure 1 the real part is 0 (there is no cosine part) and for figure 2 the imaginary part is 0 (there is no sine part). Where the phase offset is not a multiple of π then the complex number will have both a real and imaginary part. The relationship between real and imaginary parts and the phase and magnitude of complex numbers is shown in the figure below.



$$magnitude = \sqrt{(real)^2 + (imag)^2}$$
$$\phi = \arctan(imag / real)$$

Figure 3: derivation of phase and magnitude from real and imaginary components

For a sinusoidal function the ideal magnitude response (magnitude plotted against frequency) should appear as a single spectral line at the frequency of the sinusoid as shown in the figure below. However the **basis functions** (the signal components which our signal is decomposed into) of the DFT are sinusoids of infinite duration as we can see from the time domain plot on the right.

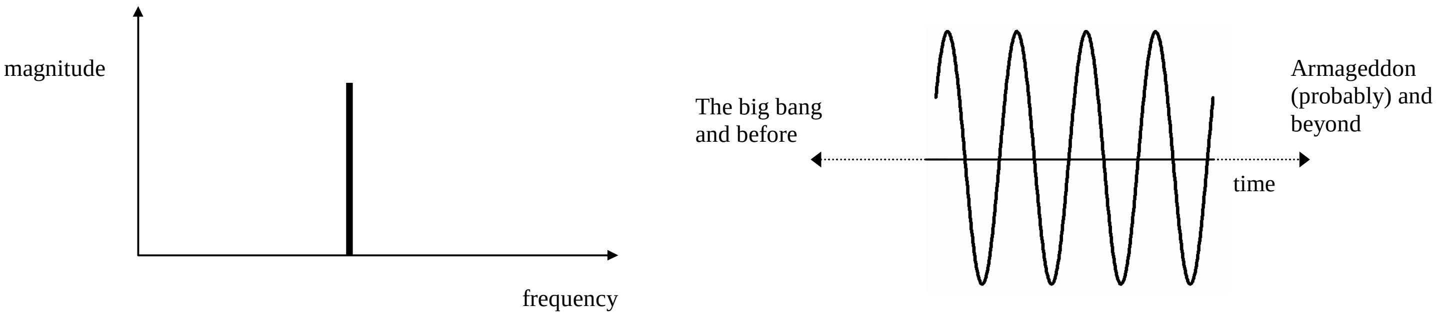


Figure 4: the magnitude response of a sinusoid of infinite duration and its time domain plot

Even if we assume that the components of an audio signal are stationary (i.e. the parameters of its components, amplitude and frequency, do not change) throughout its duration and we take a single DFT of that signal, we will rarely obtain a magnitude response for a single sinusoid like the one in the above figure; only if we use a rectangular window whose length is an *exact* integer mulitple of the pitch period will we get a single non-zero point in the magnitude response. Even if we could obtain such a response, stationary audio signals convey very little information so we are usually concerned with analysis of non-stationary ones – the parameters will vary over time as a result of changing pitch, timbre, and dynamics as well as vibrato, tremolo and so on. In order to track these changes to the signal that happen over time we need to divide our signal into small sections (frames) and perform a DFT on each one. We need short frames to properly track changes in time, but the shorter our frames (and hence our DFT lengths) are, the further away from the ideal length (infinite) they are and so the magnitude response of our sinusoid becomes more smeared in frequency. When we divide a signal into shorter frames and perform analysis by DFT on each frame this process is known as the short-time Fourier transform (STFT).

In order to smooth our frames so that they appear to fade from and back to infinity we usually apply a tapered window to each frame and overlap these frames so that the overall level of our signal does not undulate over time. Even if we do not apply a smoothing window, the process of dividing the signal up into frames produces rectangular windows of the signal (this is usually undesirable since a rectangular window produces worse smearing in the frequency domain than a window with a smoother shape). The figure below shows a single frame of a signal after a Hann window has been applied. The tapered result is the closest we can get to an infinitely long, stationary sinusoid.

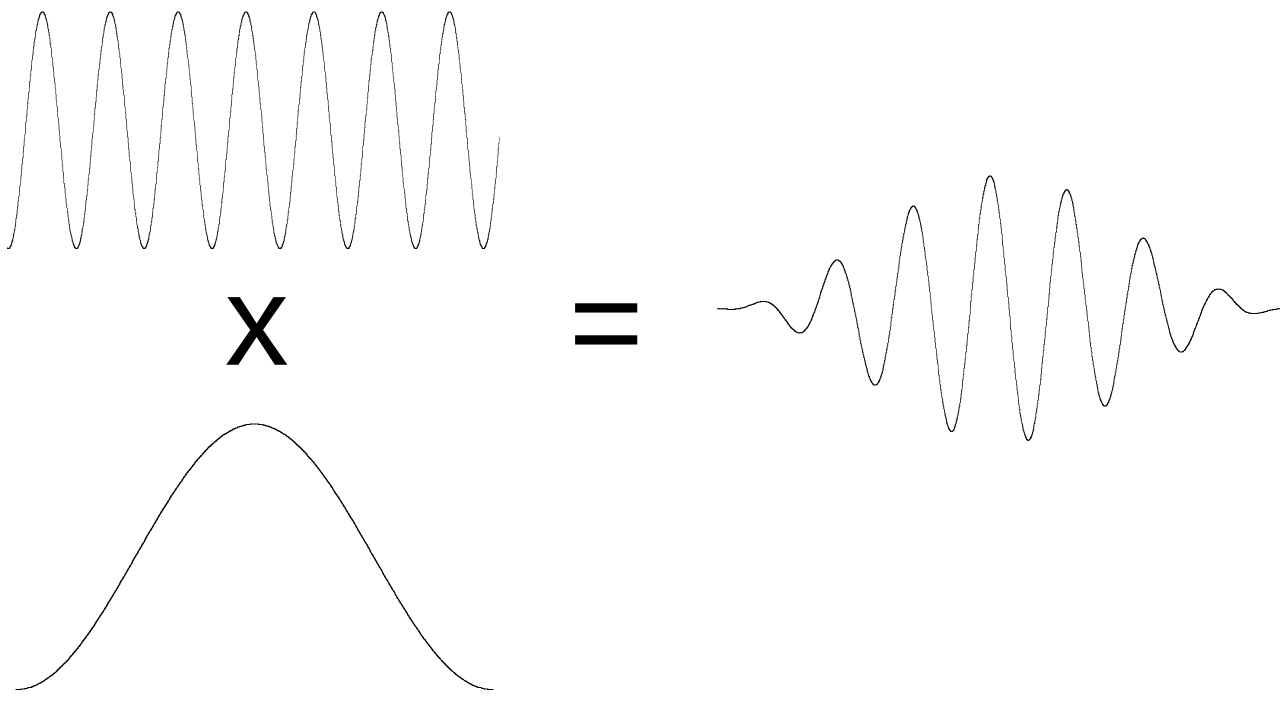


Figure 5: Multiplying one frame of a sinusoidal signal by the Hann window

The modulation property of the DFT holds that multiplication in the time domain (as for our windowing operation in the figure 5) is equivalent to convolution in the frequency domain[3]. So the representation (magnitude response) of the tapered sine wave in figure 5 will be the magnitude response of the infinitely long sinusoid in figure 4 convolved with that of the Hann window (shown below)[4].

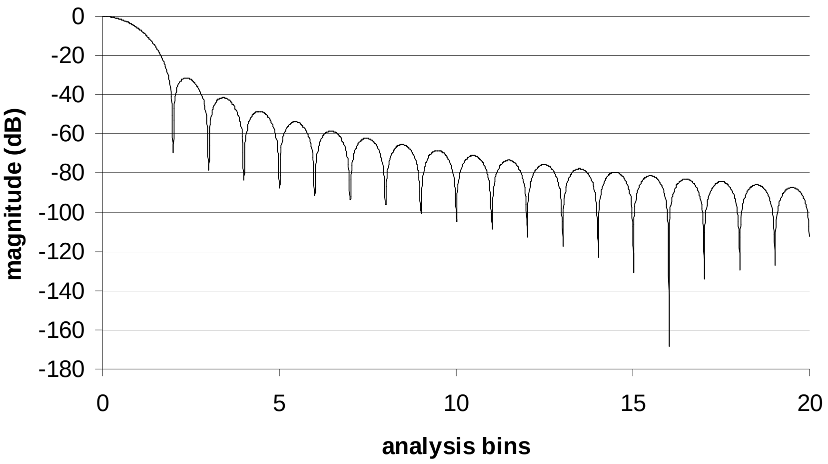


Figure 6: Magnitude response of the Hann window (0dB = peak of response, centre of bin 0)

The magnitude spectrum of this convolution for a 1 kHz sine wave analysed with a 1024 window and an 8 times zero-padded DFT is shown below. The response is slightly lopsided because 1 kHz does not coincide with the exact centre of an analysis bin and so the result of the convolution is not symmetrical around the frequency of the sinusoid.

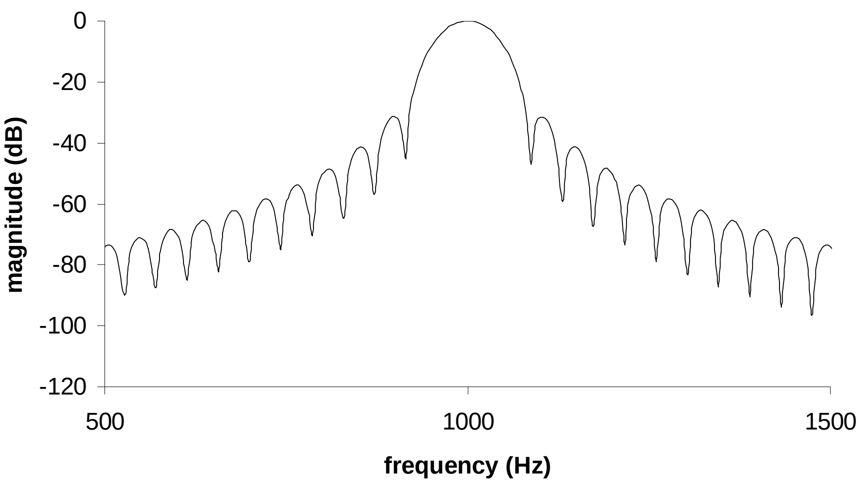


Figure 7: Magnitude response of the Hann window convolved with a 1 kHz sinusoid

So, provided we know the magnitude response of our windowing function, we can predict what a sinusoidal function will look like when we analyse it with that window. Clearly the magnitude response is at its maximum for the bin in which the sinusoid resides so one of the most straightforward ways of identifying a sinusoid is by searching for maxima in the magnitude response, and this is commonly the first step in sinusoidal identification. The relative magnitudes of surrounding bins can be used as a measure of how close a peak in the spectrum is to that of a stationary sinusoid. This becomes more complicated for a spectrum containing closely spaced sinusoids since more than one sinusoid will exert an influence over the same bin. When analysing peaks it is common to consider those bins either side of the peak up to minima either side as a spectral region[5]. The MPEG Layers I and II use a simple measure that considers the relative magnitude of the peak bin and close neighbours to determine whether a peak represents a sinusoid. A peak is considered to be a sinusoid if the following condition is met, where $X(k)$ and $X(k+j)$ are magnitude components of a 512 sample analysis frame:

$$X(k) - X(k+j) \geq 7dB$$

For MPEG layer 1, j is chosen as follows[6]:

If $2 < k < 63$ then *j*= -2 and 2.
If $63 \le k < 127$ then *j*= -3, -2, 2 and 3.
If $127 \le k < 250$ then *j*= -6, -3, -2, 2, 3 and 6.

This smearing of sinusoids in frequency due to windowing effects has other uses. The frequency shaping employed in the *Shapee* cross synthesis algorithm relies on the effects of windowing. Partials from one signal modify spectral regions of another signal because they have been spread in frequency. If this were not the case then unless partials in both sounds coincided in very narrow frequency regions (the width of one analysis bin) they would not be able to exert any influence over each other[7].

So far we have considered time and frequency domain representations of stationary sinusoids. One of the assumptions of the STFT is that the signal being analysed is stationary for the duration of each analysis frame. Many signals, such as those with vibrato or tremolo for example, have continuously varying frequency and/or amplitude and sinusoids which exhibit such behaviour have different magnitude responses. Modulation has the effect of flattening the main lobe of the analysed sinusoid. The figure below shows the main lobes for a stationary sinusoid and one whose frequency is increasing linearly at approximately 43 Hz and whose amplitude is falling linearly at 2 dB per frame for a 1024 sample, 32 times zero padded DFT (sinusoid sampled at 44.1 kHz). As well as assisting us in identifying non-stationary sinusoids, knowledge of this change in shape of the spectral peak is important for estimating the amplitude of such signal components as we shall see shortly.

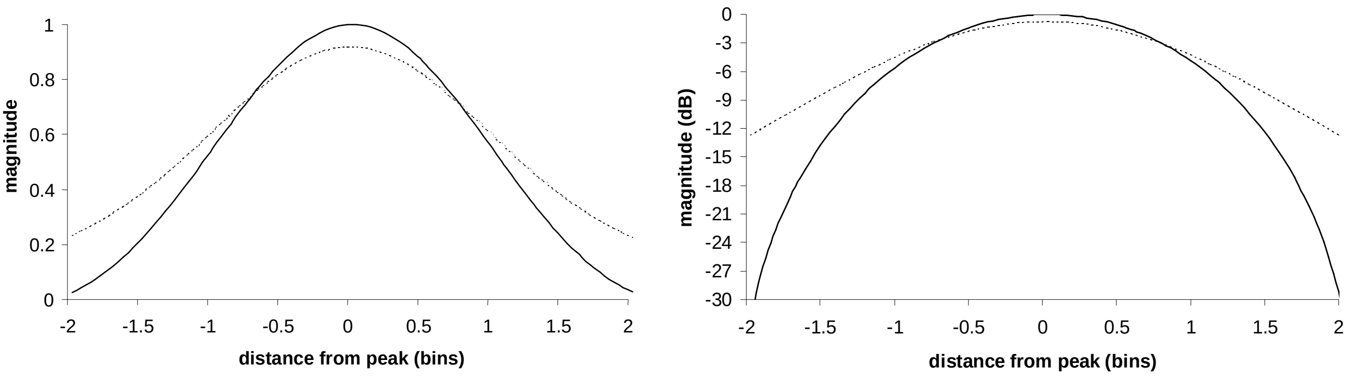


Figure 8: Main lobe of stationary (solid line) and non-stationary (dotted line) sinusoids as a function of absolute magnitude (scaled to peak of stationary sinusoid) and in decibels (relative to peak of stationary sinusoid).

Given knowledge of the analysis window a 'sinusoidality' measure can be applied to a peak in the spectrum and its surrounding bins. This is a measure of the correlation between the actual bin magnitudes surrounding the peak and the window function shifted to the estimated frequency:

$$\Gamma_{peak} = \left| \sum_{f_{peak}+B}^{f_{peak}+B} H(f).W(f) \right|$$

H(*f*) is the measured and normalised DFT and *W*(*f*) is the shifted and normalised window function. *B* is the half bandwidth over which the correlation is measured (often the bandwidth of the main lobe of the window function) and so the number of points considered in the correlation measure depends on the degree of zero padding used in the DFT. Amplitude and frequency modulation effects can be accounted for by suppressing the modulation or by estimating it[8] and adapting the window function W(*f*) accordingly[6].

Now that we have some knowledge of what sinusoids look like in the frequency domain and we have discussed some techniques for identifying them in the spectrum of a signal we can now consider methods for estimating their parameters.

Estimating frequency

If we consider a 1024 point DFT of a signal sampled at 44.1 kHz we can calculate its frequency resolution by dividing the sampling rate of the signal by the DFT size. Therefore the size of each analysis bin, or the frequency resolution of our analysis (the narrower the better), is approximately 43 Hz. Since we are only concerned with the range of frequencies below the Nyquist limit (half the sampling rate of our signal) we only have, in effect, 513 analysis bins of information. This is because the DFT of real (e.g. audio) data has 'complex conjugate symmetry' meaning that we can extract all the information from half of the analysis bins plus the zeroth bin. We cannot differentiate between two sinusoids that are closer together in frequency than 43 Hz, regardless of the window we use. A 2048 point DFT of the same signal will have double the frequency resolution (approximately 21.5) Hz but with a halving of the time resolution, since we are now analysing a frame of twice the length.

In order to understand the relationship between time and frequency resolution it is helpful to consider two sinusoids one of frequency 1000 Hz and another of frequency 1001 Hz. In order to distinguish between these two sinusoids in frequency we need to wait until the difference in cycles between them is at least one. With a difference in frequency of 1 Hz a one cycle difference will take 1 second to occur. Therefore the length of our analysis frame must be at least 1 second (for a rectangular window) in order to make the distinction between the two. Any changes to our two sinusoids *intra*-frame (i.e. within this single analysis frame) will be blurred together. If the difference between the two is 2 Hz then we need an analysis frame of half that length ($1/f = 0.5$) in order to make the distinction. It doesn't matter whether the two frequencies are 1 Hz and 2 Hz or 10 001 Hz and 10 002 Hz, we still require a 1 second analysis frame to distinguish between them.

Given that a 1024 frame DFT of a signal sampled at 44.1 kHz is a common compromise between time and frequency resolution for spectral analysis of audio we need to find a more accurate measure of sinusoidal frequency than the centre frequency of an analysis bin. For example a sinusoid of frequency 64 Hz will produce a peak in an analysis bin with a centre frequency of 43 Hz. If we use this as an estimate of frequency our error will be roughly an interval of a fifth, a significant error in musical terms! However, we can use the magnitude of the peak and that of adjacent bins, or its phase and that of adjacent frames for the same bin, to improve our estimates for stationary sinusoids significantly.

The term **phase vocoder** describes a system that performs an STFT on an input signal and derives magnitude and phase values for each analysis bin in each frame. It is the derivation of phase information in addition to magnitude that distinguishes it from the **channel vocoder**. The use of the word **vocoder** (voice–coder) is anachronistic since vocoders were first developed for speech analysis and synthesis but are now applied to many other signal types. It is interesting to note that, although many applications of time-frequency analysis such as time modification independent of pitch (e.g. time stretching), require analysis and manipulation of phase to achieve high quality results[9] many high quality modifications, such as the frequency shaping cross-synthesis of *Shapee* can be performed by simply retaining phase values for one sound and 'grafting on' the maginitude date of another[7].

If we consider a sinusoid whose frequency is equal to the centre of an analysis bin we can see from the figure below that if we divide it into consecutive analysis frames then its phase offset in each frame is the same.

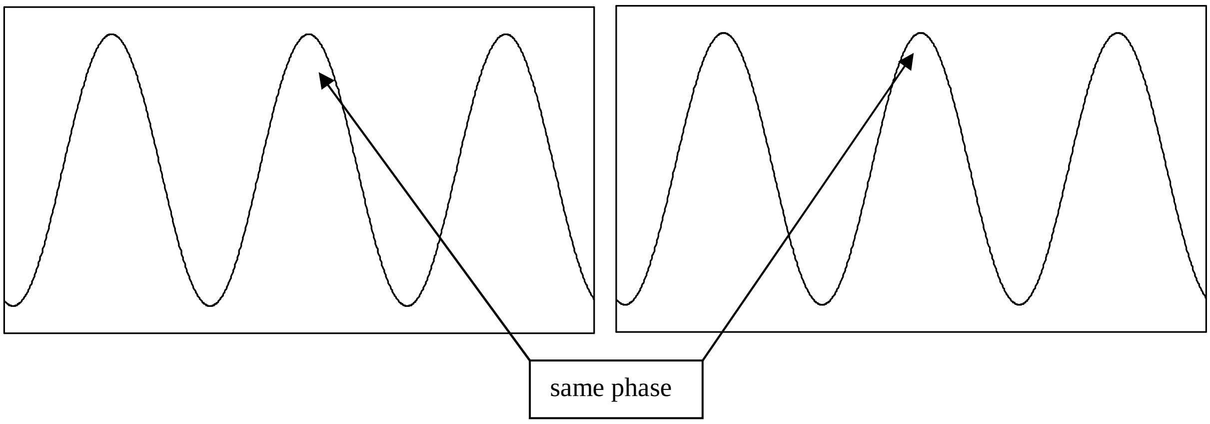


Figure 9: Successive analysis frames for sinusoid whose period matches centre frequency of analysis bin

Therefore for each successive analysis frame the measured phase of the peak bin will be the same. For our DFT parameters (described above) the peak bin for this sinusoid has a centre frequency of 129 Hz and, since there is no change in phase between successive frames, we can take this as an accurate measure of frequency. It is important to remember here that we are not considering overlapping frames in this example. Although, as discussed earlier overlapping frames are desirable for windowed signals to prevent amplitude modulation of the signal according to the shape of the window function, this example is more straightforward if we consider the non-overlapping case.

If we now consider a sinusoid whose frequency falls above the centre frequency of this bin (not quite at the halfway point between the centre frequency of this bin and the one above it) then we can see that the phase offset between successive frames is not the same. In this case we use the phase difference between frames to calculate the deviation in frequency of the sinusoid from the centre frequency of the bin. The centre frequency is calculated by multiplying the bin number by the width (in Hz) of the bin (the sample rate of the signal divided by the frame length).

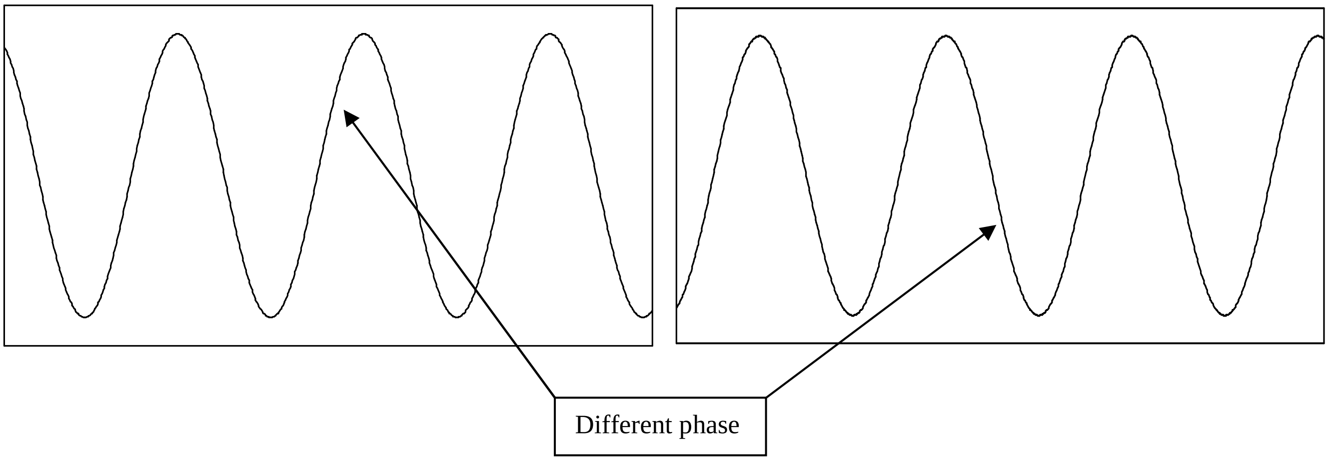


Figure 10: Successive analysis frames for sinusoid whose period matches centre frequency of analysis bin

We have no way of discriminating between multiples of 2π (since 2π represents a complete rotation around the origin in figure 3) but for the peak bin this is enough, as a phase difference of $\pm\pi$ represents the range of frequency deviation for the whole of a single bin (half a bin width above and half a bin width below the centre of the bin). A simple expression of the calculation of frequency based on inter-frame phase differences is:

$$f_{sinusoid} = B(n \pm \frac{\phi_{difference}}{2\pi})$$

Where *B* is the bin width (in Hz or radians according to preference), *n* is the bin number and $\phi_{difference}$ is the phase difference between two successive frames. With no frame overlapping in our analysis we can obtain a value in the correct range for the peak bin but for the two adjacent bins the deviation value could be in the range of $\pm 1.5B$ and this range increases as we move away from the peak. Such deviations could lead to phase offsets in the range of $\pm 3\pi$ which will lead to incorrect deviation measurements (for example frequency offsets of 0.5*B*, 1.0*B* and 1.5*B* will each give a phase offset of π). Such incorrect deviation measurements will lead to alias frequencies appearing in our analysis. If we overlap by a factor of 4 then frequency deviations of 0.5*B*, 1.0*B* and 1.5*B* will give phase offsets of $\pi/4$, $\pi/2$ and $3\pi/4$ (since the phase only has a quarter of the time to increment) which are unambiguous. As well as being a requirement to prevent the overall signal level undulating with the window shape, overlapping windows **over-sample** the spectrum and so offer control over aliasing distortion around sinusoidal peaks.

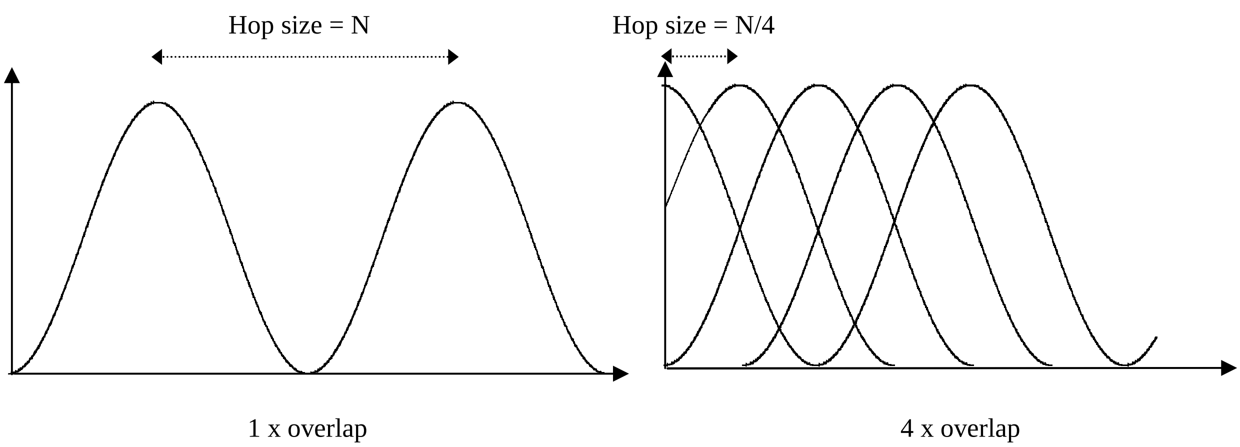


Figure 11: different overlaps

The greater the overlap employed in the time domain, the greater the range of bins either side of a sinusoidal peak that give a correct estimate of the frequency of that sinusoid. The table overleaf shows estimates for the peak bin and its eight closest neighbours for a stationary sinusoid of 1 kHz with for overlaps of 1x and 4x.

	1 x overlap		4 x overlap	
bin	**magnitude**	**frequency estimate**	**magnitude**	**frequency estimate**
peak - 4	0.4	827.7	0.4	827.7
peak – 3	0.9	870.8	0.9	827.7
peak – 2	3.0	913.9	3.0	827.7
peak – 1	43.7	956.9	43.7	1000.0
peak	123.9	1000.0	123.9	1000.0
peak + 1	85.0	1043.1	85.0	1000.0
peak + 2	6.8	1086.1	6.8	1000.0
peak + 3	1.4	1129.2	1.4	1172.3
peak + 4	0.5	1172.3	0.5	1172.3

Table 1: frequency estimates close to peak for different overlaps (figures given to 1 decimal point)

A code listing in C is given which demonstrates how frequency can be estimated from phase differences[10]. This code was used to produce the figures in table 1.

```
void FrequencyEstimation(int ComplexSize, double *phase, double *lastPhase, double *frequency)
{
        double phasediff, deviation;
        double hop = 1/overlap;
        int piSigner;

        // complexSize is the size of the array of magnitudes calculated from the complex array
        // produced by the DFT operation. The size is (N/2) + 1, where N is the DFT size.

        // phase is the calculated phase for the current frame
        // lastPhase is the calculated phase for the previous frame

        for (bin = 0; bin < ComplexSize; bin++)              {
                //phase unwrapping
                phasediff = phase[bin] - lastPhase[bin];     //phase difference
                phasediff -= (double)bin * doublePi * hop; //subtract expected phase difference

                // map the phase shift into the region +/- pi
                piSigner = phasediff/pi;
                if (piSigner >= 0) piSigner += piSigner & 1;
                else piSigner -= piSigner & 1;
                phasediff -= pi * (double)piSigner;

                // calculation deviation in fractions of a bin, add to the bin number and
                // multiply by the bin width (in Hz) to get the frequency estimate in Hz
                deviation = (overlap * phasediff)/doublePi;
                frequency[bin] = ((double)bin + deviation) * binwidth;
        }
}
```

Two approaches to estimating the frequency of sinusoids using magnitude data are those of parabolic and triangular interpolation. These rely on knowledge of the magnitude spectrum of the window at and around the peak bin to determine the precise location of a spectral peak between bins. Parabolic interpolation takes advantage of the fact that the magnitude response of most analysis windows when expressed in decibels is close in shape to that of a parabola. The following equation is used to obtain a frequency estimate using this method. The figure below illustrates this.

$$f_{sinusoid} = B\left(n + \frac{1}{2}\frac{M_{n-1} - M_{n+1}}{M_{n-1} - 2M_n + M_{n+1}}\right)$$

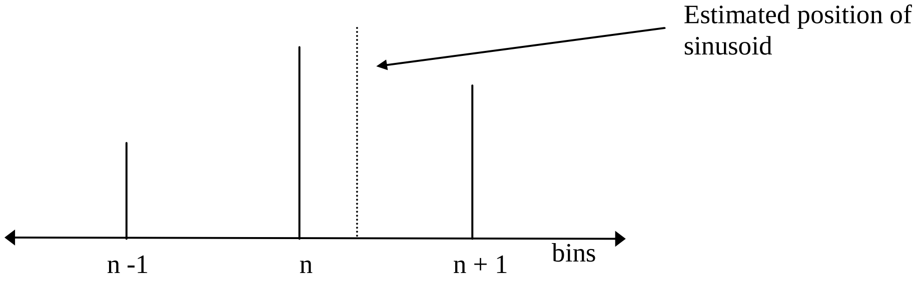


Figure 12: parabolic interpolation

Here B is the bin width, n is the peak bin and M is the magnitude of a bin expressed in dB. Using our example DFT (1024 frames, 44.1 kHz, Hann window) to analyse a 1 kHz sinusoid we obtain an estimate of 1000.6 kHz with this method. The accuracy can be improved by zero-padding the DFT and by using a specially designed window whose time domain function is the inverse transform of a function whose main lobe shape is as close as possible to that of a parabola (however this may adversely affect other aspects of window performance, such as the magnitude of side lobes which we wish to keep as low as possible).

The triangle algorithm is named after the shape of the main lobe of the window function in the frequency domain, although this is when the window is plotted with linear rather than logarithmic (dB) magnitude. After a peak has been identified, two straight lines are drawn through the bin magnitudes and the frequency estimate is taken as the point at which these two lines (which form the two opposing slopes of the triangle) intersect. The slope of the lines is determined by calculating the best fit with the least squared error[11].

Another method which uses DFT magnitudes is the derivative algorithm[12]. However this requires the computation of two DFTs for frequency estimation – one DFT is of the sampled signal (as for the other methods discussed so far), the second is of the derivative of the signal. For a sampled signal the closest approximation to the derivative of the signal is:

$$y[n] = F_s(x[n] - x[n-1])$$

where $x[n]$is the sampled signal and $y[n]$ is the approximation of the derivative. This is effectively a high pass filtering operation whose frequency dependent gain can be calculated. Therefore the difference in derivative (high pass filtered) and standard (non high pass filtered) DFT magnitudes can be used to produce an esitmate of the frequency of the sinusoid. The gain of the filter G is given by:

$$G = \frac{dM_{peak}}{M_{peak}} = 2F_s \sin\left(\frac{2\pi f_{\text{sinusoid}}}{F_s}\right)$$

therefore, if we know the gain from taking the ratio of the two DFTs then:

$$f_{\text{sinusoid}} = \frac{F_s \arcsin\left(\frac{G}{2F_s}\right)}{\pi}$$

The following C code extract gives an example implementation of this method.

```
void FrequencyEstimation(double *mag, double *Dmag, double *frequency)
{
               //frequency estimation using derivative method
               double SROverPi = SampleRate/pi;
               double RecipDoubleSR = 1/(2 * SampleRate);
               for (int bin = 0; bin < ComplexSize; bin++)          {
                      if(mag[bin] == 0.0) mag[bin] = FLT_MIN;      //to avoid divide by 0, needs
                                                                   //float.h
                      frequency[bin] = Dmag[bin]/mag[bin];
                      frequency[bin] *= RecipDoubleSR;
                      frequency[bin] = asin(frequency [bin]);
                      frequency[bin] *= SROverPi;
               }
}
```

This method takes account of phase (even though the phase from both DFTs is not used) since the difference data actually forms an overlapping frame with the original data:

Data set for single frame of DFT: x[0]……………..x[n-1]
Next frame: x[n]…………..x[2n–1]

Data set for single frame of difference DFT is calculated from: x[-1]………x[n-1]
Next frame: x[n-1]…………x[2n-1]

In fact, considering the time-shifting property of the DFT, then this method is equivalent to the phase difference method discussed earlier in this article with a hop size (distance between successive analysis frames) of one[13]. To evenly sample the spectrum with a hop size of 1 the overlap would be 1024 x with our example STFT. If we employ the derivative method with a lower overlap we are not sampling the spectrum evenly since the frequency estimate is measured between two sample periods (as the first-order difference is used). For the phase difference method our frequency estimate is averaged over the hop distance from one frame to the next giving a frequency estimate measure across the whole length of an analysis hop.

One final method for estimation is that of frequency reassignment[14]. The general method of reassignment, as well as estimating frequency deviations from the centre of analysis bins, can also be applied to the position in time of spectral data. Time reassignment provides estimates of deviations from the centre of analysis frames. Reassignment frees the time-frequency representation from the grid structure imposed by the frame length and the hop size of the STFT. Once an analysis window has been chosen, two further windows are calculated – one that is ramped in the frequency domain (for frequency reassignment) and one that is ramped in time (for time reassignment). The frequency domain window can be calculated in the time domain by calculating the first order difference of the original window (as we do for the actual signal with the derivative method discussed previously). Three example windows for reassignment are illustrated in the figure below.

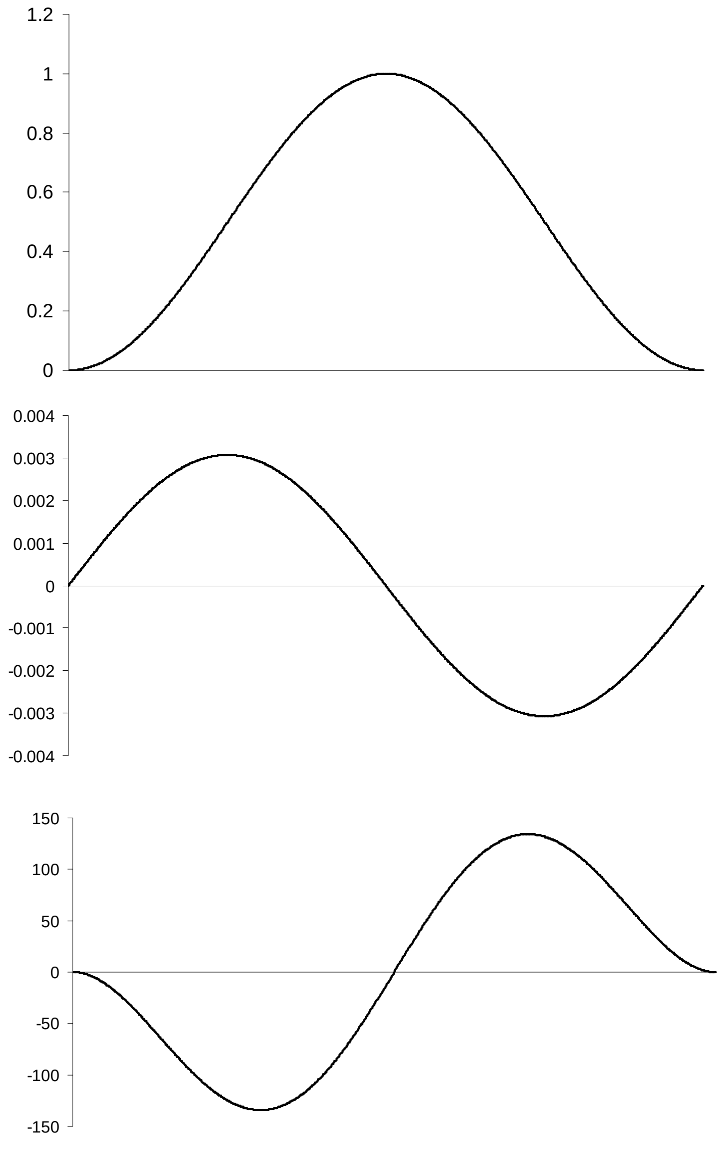


Figure 13: example windows for reassignment: Hann (top), frequency ramped Hann (middle) and time ramped Hann (bottom)

The estimate of frequency deviation (in Hz) from the centre of an analysis bin is given by:

$$-B\Im\left\{\frac{DFT_{\text{frequency ramped window}}}{DFT_{\text{standard window}}}\right\}$$

where B is, as usual, the bin width in Hz and *DFT* represents the complex value obtained for that bin by the DFT. The estimate of time deviation (in seconds) from the centre of an analysis frame is given by:

$$-\frac{1}{F_s}\Re\left\{\frac{DFT_{\text{time ramped window}}}{DFT_{\text{standard window}}}\right\}$$

where F_s is the sampling rate of the signal. Time reassignment can be used for sharpening of transients in sinusoidal analysis which reduces the 'smearing' of attack portions (a sudden change in level within a frame is averaged across the whole frame by DFT analysis)[15]. C code for frequency reassignment is presented below (unlike previous code examples window generation is also included here since it forms an important part of the method):

```
void CreateWindows(double *HannWindow, double *ReassignFrequencyHannWindow, int FrameSize)
{
        double angle, temp;
        double multiplier = doublePi/(double)(FrameSize + 1)

        //generate Hannwindow
        for (a = 0; a < size; a++)    {
                angle = multiplier * ((double)a + 1.0);
                temp = 0.5 * cos(angle);
                HannWindow[a] = 0.5 - temp;
        }

        //this will calculate derivative of any window
        //provided the window is symmetric
        double stepHeight = (HannWindow[0] + HannWindow[size-1]) * 0.5;
        double* tempArray;
        tempArray = new double[size + 2];
        tempArray[0] = tempArray[size + 1] = 0;
        for (a = 1; a <= size; a++) tempArray[a] = HannWindow[a-1] - stepHeight;
        for (a = 0; a < size; a++) ReassignFreqWindow[a] = (tempArray[a+2] - tempArray[a]) * 0.5;
        ReassignFreqWindow[0] += stepHeight;
        ReassignFreqWindow[size-1] -= stepHeight;
}

void FrequencyEstimation(double *ComplexBuffer, double *ComplexFreqBuffer, double *Frequency, double
BinWidth)
{
        //ComplexBuffer is complex output of DFT on Hann windowed data
        //ComplexFreqBuffer is complex output of DFT on 'frequency weighted Hann' windowed data
        double denominator, deviation, imgainaryNumerator, temp;
        DCplx conjugate;

        //calculate complex conjugate for complex division
        conjugate.re = ComplexBuffer[bin].re;
        conjugate.im = 0.0 - ComplexBuffer[bin].im;

        //do complex division (only the imaginary part is required for frequency reassignment)
        denominator = pow(ComplexBuf.re, 2) + pow(ComplexBuf.im, 2);
        numerator.im = (ComplexFreqBuffer[bin].re * ComplexBuffer[bin].im)
                + (ComplexFreqBuffer[bin].im * ComplexBuffer[bin].re);

        //calculate offset
        deviation = imaginaryNumerator/denominator;

        //calculate frequency offset in Hz
        Frequency[bin = ((double)bin - deviation) * BinWidth;
}
```

For more information on the performance for different signal types of each of the five frequency estimation techniques presented here the reader is directed to published work on this subject[11,13, 16].

Correcting amplitude

We have seen how the window shape affects the magnitude of the measured DFT spectrum and how this magnitude varies with distance from the centre of an analysis bin (see figure 6 for example). This means that amplitude estimates for the underlying sinusoidal function will be incorrect unless the frequency of the sinusoid is at the centre of the bin from which the magnitude is measured. We can use knowledge of the window shape in the frequency domain and the deviation from the bin centre to correct this error. It is straight forward to extend the parabolic interpolation discussed earlier to estimate the position of the parabolic peak on the magnitude as well as the frequency axis by the equation

$$Amp_{sinusoid} = M_2 - \frac{1}{8}\frac{(M_1 - M_3)^2}{(M_1 - 2M_2 + M_3)}$$

This gives the amplitude of the sinusoid in dB.

Rather than taking the main lobe shape as being a parabola when its magnitude response is expressed in dB we can calculate the power spectrum of the window (either by direct evaluation or by storing the magnitude coefficients of a zero-padded DFT in a look-up table[12]). The magnitude spectrum of the Hann window can be directly evaluated and C code for amplitude correction using this method is presented below along with the equation for the magnitude response (d is the offset, in bins, from the main lobe peak)[17]. Note that for this method we are concerned with the magnitude itself and not a logarithmic function of it.

```
double AmplitudeCorrection(double Frequency, double Mag, double BinWidth
{
        double Deviation, DeviationSquared, Scaler;

        //Frequency is frequency estimate
        //Mag is peak bin magnitude
        //BinWidth is width of analysis bin in Hz

        //calculate deviation from centre of bin
        Deviation = Frequency/BinWidth;
        Deviation = Deviation - (int)Deviation;
        if (Deviance > 0.5) Deviation = 1 - Deviation;

        DeviationSquared = Deviation * Deviation;
        if(Deviation == 0.0) Scaler = 1.0;    //no need to adjust estimate
        else    {
                Scaler = (sin(pi * Deviance))/((doublePi * Deviance) * (1 - DeviationSquared));
                Scaler *= 2.0;
        }
        return Mag/Scaler;
}
```

$$W_{Hann}(d) = \frac{\sin(\pi d)}{2\pi d(1-d^2)} \quad , d \neq 1$$

It should be remembered that the above evaluation of the magnitude response will not hold for a non-stationary sinusoid. The next section describes how we can measure non-stationarities in sinusoids.

Estimating amplitude and frequency modulation

The effect on mainlobe shape of frequency and amplitude modulation was discussed earlier in this review. We can estimate changes in amplitude and frequency parameters of a sinusoid by calculating the differences in them across successive frames. However, for real time systems and to assist us in tracking individual sinusoids from one frame to another it is desirable to have some knowledge, intra-frame, of changes in sinusoidal parameters. This makes partial tracking more robust and allows us to consider non-stationary sinusoids when we are looking at the shape of the magnitude spectrum around a peak to determine if the underlying signal component is actually a sinusoid or not and/or to correct the amplitude estimate. If we take a DFT of an entire digital signal we have a frequency domain representation of that signal yet we have no timing information for the spectral components within that representation. However, if we take the inverse DFT of this data the signal is reconstructed perfectly so time information is not lost in the DFT, rather it is encoded in the phase relationships between analysis bins. For a stationary sinusoid the phase across bins around the peak is constant provided that zero-phase windowing is used. Empirical studies have shown that for linear frequency and exponential amplitude modulation there is a phase shift across these bins. Therefore amplitude and frequency modulations can be estimated provided the DFT is sufficiently zero-padded (8x in these studies)[18]. My own research is currently looking at new ways of estimating amplitude and frequency modulation using the reassigned STFT.

Once such modulations have been estimated they can be incorporated into our sinusoidal model, improving sinusoidal identification and estimation as well as partial tracking across frame boundaries.

Conclusion

This review has covered the main considerations when using the STFT to identify and obtain parameter estimations for sinusoidal functions. Models that require sinusoidal extraction cover a wide of audio processing applications[19, 20]. Clearly this is a large area within current research of audio analysis and modelling techniques since such descriptions of sound are often easier to engage with than the rather abstract parameters of the DFT. What has been presented here is only a fraction of knowledge and techniques in this field but hopefully it is a useful starting point.

References and notes

1. *On the Sensation of Tones* – H. Helmholtz (Dover, 1875)
2. See the appendix in the *Computer Music Tutorial* (ed. Roads) for a very clear and thorough introduction to Fourier analysis in general and this kind of mathematical notation.
3. Chapters 2 and 3 of *Introductory Digital Signal Processing with Computer Applications* by Lynn and Fuerst give an excellent introduction to time and frequency domain analysis covering topics such as this.
4. This interpolated plot of the magnitude response of the Hann window was produced by taking a 64 times zero padded DFT of a 1024 point Hann window. The zero padding provides values for the response at intervals of 1\64 of a bin rather than just the one value per bin that would have been obtained if zero padding had not been used.
5. *Analysis of Reassigned Spectrograms for Musical Transcription* – S. Hainsworth, M. Macleod and P. Wolfe (Proceedings of ICASSP-01)
6. *Sinusoidal Parameter Extraction and Component Selection in a Non-Stationary Model* – Lagrange, Marchand and Rault (Proceedings of DAFx-02)
7. *Frequency Shaping of Audio Signals* – Christopher Penrose (ICMC Proceedings 2001). My own real time implementation of Shapee for VST-PC is available in the 'computer music tools' section of www.jezwells.org.
8. *Signal Characterisation in terms of Sinusoidal and Non-Sinusoidal Components* – G. Peeters and X. Rodet (DAFx-98).
9. *About This Phasiness Business* – M. Dolson and J. Laroche (ICMC Proceedings 1997)
10. This is adapted from code by Stephan Sprenger which can be found in his *Pitch Scaling Using the Fourier Transform* which is a very useful introduction to using the STFT for audio analysis and transformation. This and other articles are available at www.dspdimension.com
11. *Survey on Extraction of Sinusoids in Stationary Sounds* – F. Keiler and S. Marchand (Proceedings of DAFx-02).
12. *High-Precision of Fourier Analysis of Sounds Using Signal Derivatives* – M. Desainte-Catherine and S. Marchand (Journal of the Audio Engineering Society, 7/8, 2000)
13. *On Sinusoidal Parameter Estimation* – S. Hainsworth and M. Macleod (Proceedings of DAFx-03).
14. *Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method,* F. Auger and P. Flandrin, *(*IEEE Transactions on Signal Processing, vol. 43).
15. *On the Use of Time-Frequency Reassignment in Additive Sound Modeling* – K. Fitz and L. Haken (Journal of the Audio Engineering Society, 11, 2002).
16. *Time Frequency Reassignment: A Review and Analysis* – S. Hainsworth and M.Macleod (Cambridge University Engineering Report – CUED/F-INFENG/TR.459).
17. *Some Windows with Very Good Sidelobe Behaviour* – J. Nuttall (IEEE Transactions on Acoustics, Speech and Signal Processing, 1, 1981)
18. *Identification of Nonstationary Audio Signals Using the FFT, with application to Analysis-based Synthesis of Sound* – P. Masri and A. Bateman (Proceedings of IEE Colloquium on "Audio Engineering", May 1995)
19. *Sound Transformations based on the SMS High Level Attributes* – X. Serra and J.Bonada (Proceedings of DAFx-98)
20. *Real Time Spectral Expansion for Creative and Remedial Sound Transformation* – J.Wells and D. Muprhy (Proceedings of DAFx-03)