EXTRACTING SINUSOIDS FROM HARMONIC SIGNALS

Rasmus Althoff, Florian Keiler, and Udo Zölzer

Department of Signal Processing and Communications
University of the German Federal Armed Forces Hamburg
Holstenhofweg 85, D-22043 Hamburg, Germany
r.althoff@gmx.net, {florian.keiler, udo.zoelzer}@unibw-hamburg.de

ABSTRACT

This paper presents a special window function for a Fast Fourier Transform (FFT) based spectral modeling approach for signals consisting of sinusoids plus noise. The main new idea is to choose a time window function with a simple Fourier transform. With the knowledge of the Fourier transform of the window function we are able to extract the parameters (frequency, amplitude, and phase) of sinusoids in real-time with a digital signal processor.

1. INTRODUCTION

For any application that deals with power spectrum estimation and harmonic analysis, it is important to extract the line spectrum components before performing any spectral noise "smoothing" because otherwise the lines would lose their sharpness [1]. Also in speech and audio coders, an important task is to extract harmonic signals and to calculate masking thresholds for adaptive bit allocation.

Many approaches [2] have been proposed for this task, e.g. time-domain Prony's method, subband modeling, least-square fitting [3] and frequency domain methods. For the various methods the accuracy and the computational complexity differs significantly. The algorithm presented in [4] is based on the FFT and offers moderate complexity. We will use this algorithm as a reference and introduce a new robust harmonic analysis algorithm, which is fast and performs better on extracting noisy sines. It is also a block-based approach and uses the FFT. First we will discuss the analysis algorithm, explain the synthesis algorithm for the harmonic components and then present a comparison and some simulation results.

The main analysis/synthesis structure is shown in Figure 1. The input signal $s(n)$ is divided into blocks of length N and after weighting by a window, an N-point FFT is performed. The FFT length N should be chosen that the bin frequency $f_b = f_S/N$ is about $f_b \approx 20 \ldots 50$ Hz; f_S denotes the sampling frequency. Magnitude maxima ("peaks") are searched in the obtained spectrum. Applying an appropriate algorithm, estimates of frequency, amplitude, and phase of corresponding superimposed sinusoids are calculated. After performing a subsequent peak tracking only those sinusoids are used which belong to the "real" signal, i.e. which are not part of the noise. The spectrum is reconstructed using the extracted data (frequency, amplitude, phase), and applying an IFFT (inverse FFT) yields the deterministic part $s_{det}(n)$ of the signal. The described structure is similar to the one used in [5], but we are using a constant window.

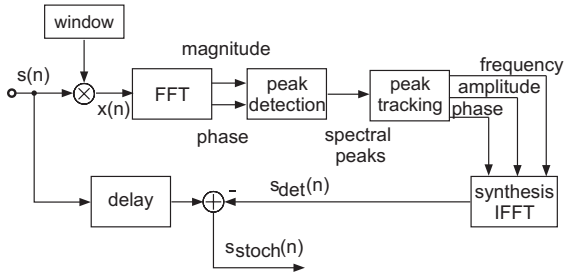

Figure 1: Main structure block diagram.

2. ANALYSIS

We present a novel algorithm to calculate frequency, amplitude, and phase of sinusoids from the FFT data. The estimated sine frequency is much more exact than just taking the bin frequency of the FFT magnitude maximum. For comparison another already proposed algorithm is described.

2.1. Triangle Algorithm

Our algorithm is named after the shape of the analysis window in the frequency domain. The idea is to use a window whose magnitude response can be determined by a simple function. Since the window is shifted by the sine frequency when applied to a sinusoid, the parameters of the function can be calculated from the FFT data. We choose here a triangular frequency response which can be described by two lines.

2.1.1. Choice of the Window

The absolute values of the frequency response should follow a triangle function; we choose the zero-phase frequency response[1]

$$A(e^{j\Omega}) = \begin{cases} 1 - \left|\frac{\Omega}{\Omega_c}\right| & , \ |\Omega| < \Omega_c \\ 0 & , \ \text{otherwise} \end{cases} \tag{1}$$

for $|\Omega| < \pi$ where $A(e^{j\Omega})$ is 2π-periodic, see Figure 2(a). Because of the desired even symmetry of the window in the time domain this results in a non-causal time window with an odd number of taps. With the even FFT length of N we get a $N-1$ tap window. Delaying $a(n) \circ\!\!-\!\!\bullet A(e^{j\Omega})$ by $N/2$ samples yields the window $w(n) = a(n - N/2)$ with $w(0) = 0$. Figure 2(b) shows

[1] $\Omega = 2\pi f T_S$ denotes the normalized frequency with the sampling frequency $f_S = 1/T_S$.

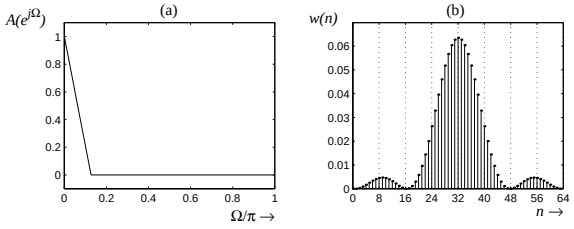

Figure 2: Triangular amplitude response (a) and corresponding causal window (b) for $N = 64$.

$w(n)$ for $N = 64$ and $\Omega_c = \pi/8$. Applying an N-point FFT (evaluation at frequencies $\Omega_k = \frac{2\pi}{N}k$) to $w(n)$ gives

$$W_d(k) = W(e^{j\Omega_k}) = e^{-j\Omega_k N/2} A(e^{j\Omega_k}) = (-1)^k A_d(k). \quad (2)$$

Thus, the values $W_d(k)$ are real valued and the sign alters. Analyzing the input signal $s(n) = \cos(\Omega_0 n + \varphi)$ results in the windowed frame ($0 \le n \le N-1$)

$$x(n) = s(n) \cdot w(n) = w(n) \frac{1}{2} \left[e^{j(\Omega_0 n + \varphi)} + e^{-j(\Omega_0 n + \varphi)} \right]. \quad (3)$$

Assuming that the spectra $A(e^{j(\Omega - \Omega_0)})$ and $A(e^{j(\Omega + \Omega_0)})$ do not overlap, we get for $0 \le k \le N/2$ the FFT of $x(n)$ according to

$$X_d(k) = \frac{1}{2} e^{j(\varphi + \Omega_0 N/2)} (-1)^k A(e^{j(\Omega_k - \Omega_0)}). \quad (4)$$

Hence the phase of $X_d(k)$ at frequency index k is given by

$$\phi(k) = \varphi + \Omega_0 N/2 + k\pi. \quad (5)$$

The choice of Ω_c (see Eq. (1)) influences the properties of the algorithm: for greater values of Ω_c noisy sines can be detected better while the frequency resolution is higher for a small Ω_c. In the discrete-frequency domain we choose the length of the triangle basis to $D = 8$, i.e. $A_d(k) = 0$ for $4 \le k \le N-4$, which has shown to be a good compromise.

The triangle is described by two lines, $h_1(k)$ in the left half and $h_2(k)$ in the right one as expressed by Equations (6) and (7).

$$h_1(k) = a \cdot k + b \quad \text{with} \quad a > 0 \quad (6)$$

$$h_2(k) = -a(k - D) - b \quad (7)$$

Notice that here the parameter k is not restricted to be an integer; the gradient of the lines is a or $-a$, respectively.

2.1.2. Algorithm Overview

The algorithm works as follows.

1. The input signal $s(n)$ is multiplied by the N-tap window $w(n)$. The resulting $x(n)$ is transformed to the frequency domain using an N-point FFT. Since the time-domain signal $x(n)$ is real-valued, the FFT can be performed using an $N/2$-point complex-valued FFT. The magnitude values are computed by $X_m(k) = |X_d(k)|$.
2. It is searched for a frequency index k_m with

$$\begin{aligned} X_m(k_m - 2) < X_m(k_m - 1) < X_m(k_m), \\ X_m(k_m + 2) < X_m(k_m + 1) < X_m(k_m). \end{aligned} \quad (8)$$

   The FFT magnitude $X_m(k)$ has therefore a local maximum at the frequency index k_m.
3. The phase is evaluated. According to Equation (5) first the corrected phase ($k \in \{k_m - 1, k_m, k_m + 1\}$)

$$\phi_c(k) = \begin{cases} \phi(k) & , \ k \text{ even} \\ \phi(k) - \pi & , \ k \text{ odd} \end{cases} \quad (9)$$

   is computed where $\phi(k) \in [0, 2\pi)$ is assumed. The phase $\bar{\phi}$ is the obtained by taking the mean value of $\phi_c(k_m - 1)$, $\phi_c(k_m)$, and $\phi_c(k_m + 1)$. Notice that $\bar{\phi}$ is the phase at time index $n = N/2$.
4. If the deviations $|\phi_c(k_m + i) - \bar{\phi}|$, $i \in \{-1, 0, 1\}$, are beyond a defined threshold, the calculation of the triangle parameters is performed (step 5), otherwise in step 2 the search for other FFT maxima is continued.
5. From the six highest spectral values surrounding $X_m(k_m)$ the parameters a and b for Equations (6) and (7) are calculated by minimizing the squared error. The peak of the resulting triangle is located at

$$k_0 = \frac{D}{2} - \frac{b}{a} = 2 - \frac{b}{a}. \quad (10)$$

   In the current implementation the amplitude evaluation is performed by calculating the energy in the frequency domain. The squared values of the obtained triangle at the six considered points are accumulated and the obtained sum is scaled. The scaling is necessary, because only three of four points of each line are used to reduce the influence of noise. Since this gives a systematic error depending on the position of k_0 between two frequency bins, a subsequent error correction is performed. The amplitude could also be obtained by calculating

$$h_1(k_0) = h_2(k_0) = D/2 \cdot a. \quad (11)$$

   An appropriate error correction has to be performed in this case as well, since Eq. (11) does not give the correct value, as it will be seen in the example shown in Figure 3(b).
6. Continue in step 2 and search for other FFT maxima.

2.1.3. Example

Fig. 3 shows an example for the FFT length $N = 64$ with the input signal $s(n) = 0.6\cos(10/64 \cdot 2\pi n) + 0.8\cos(20.5/64 \cdot 2\pi n)$. Part (a) shows the magnitude spectrum (divided by N) of the input signal windowed by a rectangular window, in part (b) on the right the values of $|X_d(k)|$ ($\circ$) and the calculated triangles are shown.

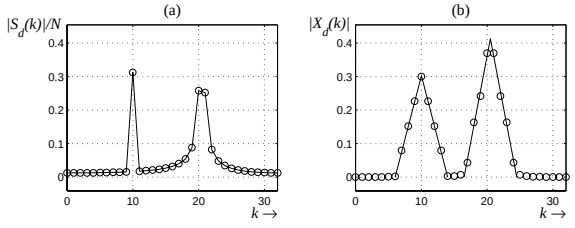

Figure 3: Triangle algorithm example for $N = 64$.

2.2. Derivative Algorithm

In the approach by S. Marchand [4] the property of continuous-time signal derivatives is exploited. The derivative of a sinusoid is again the sinusoid multiplied by the sine frequency and with a different phase, e.g. the continuous-time signal $s_c(t) = \cos(\omega_0 t)$ has the derivative $s_c'(t) = -\omega_0 \sin(\omega_0 t)$. For the discrete-time signal $s(n) = s_c(nT_S)$ the approximation $s'(n) = \frac{s(n)-s(n-1)}{T_S} \neq s_c'(nT_S)$ of the derivative is used. For the z-transforms we get

$$S'(z) = f_S H(z) \cdot S(z) = f_S \cdot (1 - z^{-1}) \cdot S(z) \qquad (12)$$

with

$$|H(e^{j\Omega})| = \sqrt{2 - 2\cos(\Omega)}. \qquad (13)$$

For the evaluation of the frequency $\Omega_0 = \omega_0 / f_S$ we compute the FFTs

$$X_d(k) \quad \bullet\!\!-\!\!\circ \quad x(n) = s(n) \cdot g(n) \qquad (14)$$

$$X_d'(k) \quad \bullet\!\!-\!\!\circ \quad x'(n) = [s(n) - s(n-1)] \cdot g(n) \qquad (15)$$

with the window $g(n) \circ\!\!-\!\!\bullet\, G(e^{j\Omega})$, where $g(n) \neq 0$ for $0 \leq n < N$ and $G(e^{j0}) = 1$. For $s(n) = a \cdot \cos(\Omega_0 n)$ the Fourier transform of $x(n)$ is given by

$$X(e^{j\Omega}) = \frac{a}{2} G(e^{j(\Omega-\Omega_0)}) + \frac{a}{2} G(e^{j(\Omega+\Omega_0)}). \qquad (16)$$

$x'(n)$ is according to Eq. (15) a superposition of two sines with different phases windowed by $g(n)$. A first approximation of Ω_0 is $\Omega_k = 2k\pi/N$; k is the frequency index where both $|X_d(k)|$ and $|X_d'(k)|$ have maxima. Assuming $|G(e^{j(\Omega_k+\Omega_0)})| \approx 0$ we get

$$X_d'(k) \;=\; X'(e^{j\Omega_k}) \approx \frac{a}{2}(1 - e^{-j\Omega_0}) G(e^{j(\Omega_k-\Omega_0)}) \qquad (17)$$

$$\tilde{\Omega}_0 \;=\; \frac{|X_d'(k)|}{|X_d(k)|} \approx \sqrt{2 - 2\cos(\Omega_0)} \qquad (18)$$

$$\Omega_0 \;=\; \arccos(1 - \tilde{\Omega}_0^2/2). \qquad (19)$$

The amplitude of the sine is obtained by

$$a = 2\frac{|X_d(k)|}{|G(e^{j(\Omega_k-\Omega_0)})|}\,. \qquad (20)$$

3. SYNTHESIS

The aim of the synthesis stage is to create a time-domain signal which is the superposition of sinusoids with amplitudes, frequencies, and phases computed in the analysis stage. We are performing the synthesis using an IFFT. Thus, we have to calculate the values in the frequency domain which have to be transformed to the time domain. For a real-valued time signal $\hat{s}(n)$ only the values $\hat{S}_d(k)$ for $0 \leq k \leq N/2$ have to be known since

$$\hat{S}_d(k) = \hat{S}_d^*(N-k) \qquad (21)$$

is satisfied (the superscript * denotes complex conjugation). Furthermore the N-point IFFT can be performed using an $N/2$-point IFFT; the samples of $\hat{s}(n)$ can then be extracted from the real and imaginary part of the result.

A frame of a synthesized sinusoid is the sinusoid windowed by a rectangular window $r(n)$. $r(n)$ is equal to one for $0 \leq n < N$ and zero otherwise. Its Fourier transform is given by

$$R(e^{j\Omega}) = e^{-j\Omega\frac{N-1}{2}} \cdot \frac{\sin(\Omega N/2)}{\sin(\Omega/2)} = e^{-j\Omega\frac{N-1}{2}} \cdot \mathrm{sinc}_N(\Omega). \qquad (22)$$

If the frame $\hat{s}(n) = \cos(\Omega_0 n + \varphi) \cdot r(n)$ has to be synthesized, we have to calculate the values $\hat{S}_d(k)$ before performing the IFFT. These values can be obtained by

$$\hat{S}(e^{j\Omega}) \;=\; \frac{1}{2} e^{j\varphi} R(e^{j(\Omega-\Omega_0)}) + \frac{1}{2} e^{-j\varphi} R(e^{j(\Omega+\Omega_0)}) \qquad (23)$$

$$\hat{S}_d(k) \;=\; \frac{1}{2}(-1)^k e^{j\frac{k\pi}{N}} [e^{j\psi} f_-(k) + e^{-j\psi} f_+(k)] \qquad (24)$$

with the terms $\psi = \varphi + \frac{N-1}{2}\Omega_0$ and $f_\pm(k) = \mathrm{sinc}_N(\frac{2\pi}{N}k \pm \Omega_0)$. To reduce the computation amount, for each detected sinusoid only the "highest" spectral values are calculated, e.g. 40 values around each detected sinusoid at an FFT length of 1024. To avoid artifacts at the frame borders, overlapping frames should be analyzed and synthesized. The synthesized frames are then added after weighting with appropriate fade in/ fade out windows.

When the sinusoidal signal component has been determined ($s_{det}(n)$ in Figure 1), the noise component is obtained by $s_{stoch}(n) = s(n-L) - s_{det}(n)$. The delay by L samples of the input signal compensates the delays caused by the FFT/IFFT operations.

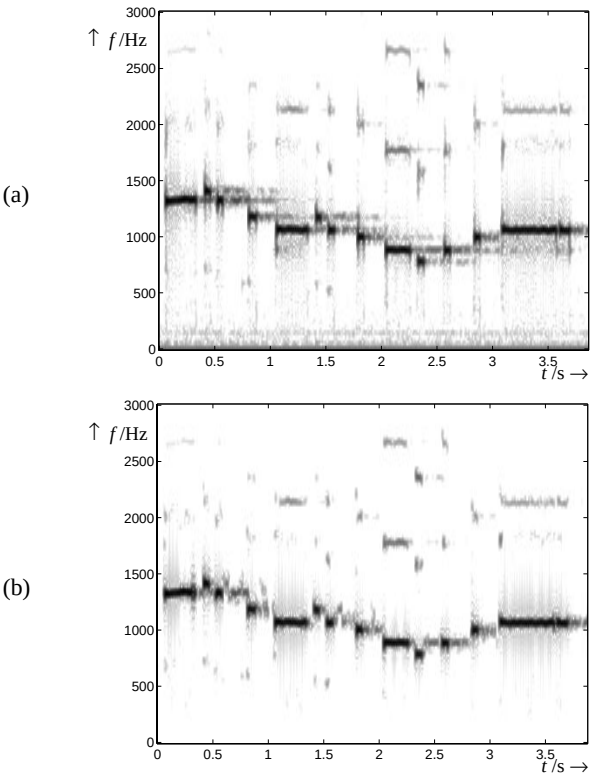

Figure 4: Time-frequency planes of tin whistle sound, original (a) and sinusoidal part extracted by using the triangle algorithm (b).

Figure 4 shows two time-frequency planes from a sample of a tin whistle sound (sampling rate $f_S = 6$ kHz). The darkness of the points corresponds to the spectrum magnitude values. The original signal is depicted in Figure 4(a) while Figure 4(b) shows the extracted sinusoidal signal, using a block length of $N = 128$ with a frame overlap of $N/2$. The triangle analysis is used and a full synthesis (use of all frequency points for each detected sine) is

applied. The only audible difference is the absence of the low-frequency component which happens to be undesired noise in this example.

4. SIMULATION RESULTS

The performance on noisy sines and the computing expense have been examined for both analysis algorithms. Both algorithms share the problem that they require a certain bandwidth for each sine, which means that two sines must have a minimum frequency difference that their spectra do not overlap (frequency resolution). For the same reason they both have a certain minimum frequency. Table 1 shows that the triangle algorithm performs better at high noise levels while the derivative algorithm is superior at low noise levels.

	adv./diasadv.
computing time	+
frequency resolution	–
miss rate (large SNR)	+
miss rate (small SNR)	–
mean frequency error	+
mean amplitude error	+
RMS-error frequency	–
RMS-error amplitude (large SNR)	–
RMS-error amplitude (small SNR)	+
max. error frequency	–
max. error amplitude	+
total error (large SNR)	–
total error (small SNR)	+

Table 1: Advantages(+) and disadvantages(–) of the triangle algorithm compared to the derivative analysis algorithm (RMS = root mean square).

There were some artifacts left; mainly the appearance of "spurious peaks". This is noise which is misclassified as a sine. The difference between a real sine tone (a piano note e.g.) and those spurious peaks is that the latter ones are much more unstable. So an algorithm is necessary which tracks the peaks in the time-frequency plane to decide which peaks are stable enough to be real sines.

4.1. Peak Tracking

We are using a peak tracking algorithm similar to the one presented in [6]. The main difference is that our algorithm considers not only stability in the terms of frequency, but also of amplitude. The algorithm is a two-step algorithm as [6], and it uses the same "birth-death-concept".

In the first step (forward search), each peak in the current frame that has not yet found a continuation in the next frame picks out the one of the following frame that matches best, i.e. among those whose frequencies do not differ more than a fixed range, the one with the least differing amplitude (in logarithmic sense) is chosen.

In the second step (backward search), all peaks in the next frame choose among those who had chosen them in the first step by minimizing the logarithmic amplitude difference. Both peaks of such a trajectory pair are marked so that they are not considered anymore.

Then, the two steps are repeated. The algorithm terminates when in step one no new possible connection is found. If in step one a peak does not find a partner in the next frame, a trajectory dies; if there are unmatched peaks in the next frame, new trajectories are "born". The lifetime is the number of frames that a trajectory exists.

The minimum lifetime of "real" sines has to be about three frames (with a bin frequency of about $f_b \approx 20 \ldots 50$ Hz). The use of this lifetime in the peak tracking algorithm avoids the "blurring" which occurs when analyzing noisy sines without peak tracking.

5. CONCLUSION

In this paper, a new approach for extracting sinusoids from harmonic signals is presented. Two different analysis algorithms and an efficient synthesis algorithm are discussed. The two analysis algorithms are compared under various aspects. For minimizing the artifacts of classifying noise as sines a peak tracking algorithm is incorporated.

The triangle analysis algorithm (without peak tracking) has been implemented on a Motorola fixed-point DSP. With an 80 MHz processor the algorithm requires 51% of the available instruction cycles, where the following settings are used: frame length $N = 1024$, frame overlapping of $N/2$, a maximum number of 102 sinusoids is modelled, and 32 frequency points around each detected sine frequency are used in the synthesis.

For detecting very low frequency sines (pitch frequency below $100, \ldots, 200$ Hz), the algorithm may be applied to a low-pass filtered and downsampled version of the input signal. For a better time resolution a longer frame overlap has to be chosen.

6. REFERENCES

[1] D.J. Thomson. Spectrum Estimation and Harmonic Analysis. In *Proc. IEEE*, volume 70, pages 1055–1096, September 1982.

[2] S.K. Mitra and J.F. Kaiser, editors. *Handbook for Digital Signal Processing*. J. Wiley & Sons, 1993.

[3] Andrew Choi. Real-Time Fundamental Frequency Estimation by Least-Square Fitting. *IEEE Transactions on Speech and Audio Processing*, 5(2):201–205, March 1997.

[4] S. Marchand. Improving Spectral Analysis Precision with Enhanced Phase Vocoder using Signal Derivatives. In *Proc. DAFX98 Digital Audio Effects Workshop*, pages 114–118, Barcelona, November 1998.

[5] X. Serra. Musical Sound Modeling with Sinusoids plus Noise. In *Musical Signal Processing*. Editors C. Roads et al, Swets & Zeitlinger Publishers, 1997.

[6] R.J. McAulay and T.F. Quatieri. Speech Analysis/ Synthesis Based on a Sinusoidal Representation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34:744–754, 1986.