A Phase Interpolation Algorithm for Sinusoidal Model Based Music Synthesis

Xiaoshu Qian

Department of Electrical Engineering
University of Rhode Island
Kingston, RI 02881, USA
qian@ele.uri.edu

Yinong Ding

DSP Research & Development Center
Texas Instruments, Inc., MS 446
Dallas, TX 75265-5474, USA
ding@hc.ti.com

Abstract

This paper presents a least square quadratic phase interpolation algorithm for sinusoidal model based music synthesis. This algorithm uses two additions with one parameter per data frame to generate the phase samples of a component sinewave. Compared with the cubic phase interpolation algorithm proposed by McAulay and Quatieri [1], the proposed algorithm is more efficient in terms of computational complexity and parameter storage. In the meantime, it also produces smoother frequency tracks (that is, with less spurious oscillations). Unlike the existing quadratic phase interpolation algorithm, where the phase measurements are ignored ("magnitude-only"), the proposed algorithm interpolates phase in a least square sense from both the phase and the frequency measurements at data frame boundaries. Thus the resulting phase samples are approximately "locked" to the measured ones. Informal listening tests on various musical instrument tones indicate that the proposed algorithm clearly outperforms the magnitude-only synthesis approach and is qualitatively comparable to the cubic one.

1 Introduction

In 1986, McAulay and Quatieri [1] of MIT Lincoln Laboratories proposed to represent a speech signal as a sum of sinusoids parameterized by time-varying amplitude, frequency and phase. Their Sinusoidal Transformation System (STS) based on this model greatly impacted the research and development of sinusoidal modeling-based music analysis and synthesis. Serra and Smith [2] of Stanford University extended the sinusoidal model to include a stochastic part in their Spectral Modeling System (SMS) for music analysis and synthesis. In both STS and SMS, the analysis and synthesis are performed on a frame by frame basis. In analysis, average amplitudes, frequencies and phases of sinusoidal components are obtained by measuring the magnitudes, frequencies and phases of spectral peaks in the Fourier transform of a frame of data. In synthesis, these parameters are interpolated to generate individual sine waves, and these sine waves are summed together to form the sinusoidal part of the sound.

Synthesizing individual sine waves in real-time imposes a major computational demand. For example, a modern professional music synthesizer typically requires a polyphony of 32. Assume each musical tone contains 40 sinusoids on average, then a total of $32 \times 40 \approx 1,200$ sinusoids needs to be generated in real-time at a sampling rate of at least 44.1 kHz. This requirement, combined with other system overhead, makes the implementation of sinusoidal model based music synthesizers extremely difficult.

Reducing this computational requirement without compromising the sound quality is our first motivation for the present work. In STS, the amplitude (in dB) and the phase track within a data frame are modeled by linear and cubic polynomials, respectively. Clearly, the computational complexity for generating sinusoidal phase samples can be reduced by using a quadratic phase polynomial instead of a cubic one. However, previous efforts in reducing the phase polynomial order have not been very successful. The main reason is that the four phase and frequency measurements at the two ends of a data frame cannot generally be made in exact agreement with a quadratic polynomial, which has only three free parameters. In the so-called "magnitude-only" synthesis approach, the phase measurements are totally ignored in favor of frequency measurements. This has been shown to cause degradation in sound quality [1]. In this paper, we propose to use a quadratic phase model, but determine the polynomial coefficients by least square fitting the model using both frequency and phase measurements. Unlike the phase track obtained from the magnitude-only algorithm, this fitted phase track will be approximately "locked" to the measured phase samples. We think the exact match of the *measured* phase and frequency with the *synthesized* ones is not necessary because the phase and frequency measurements usually contain noise. The underlying assumption is that a quadratic phase is adequate

in modeling the *true* frequency and phase. Our informal listening tests on about two dozens of musical tones confirmed this and revealed no performance degradation from the cubic phase interpolation algorithm when a quadratic phase model is used.

Another advantage of the proposed algorithm is that the resulting frequency tracks of musical tones are smoother than that obtained from the cubic interpolation algorithm. It can be shown [3, 4] that when the frequency does not change much over a data frame, which is typical for musical tones, the frequency track obtained from the cubic algorithm will *always* have slopes with opposite signs at the two ends of each data frame, displaying a "frequency track oscillation" phenomenon, as illustrated by the solid line in Figure 1. More specifically, The frequency derivatives at the frame boundaries produced from the cubic algorithm can be expressed as

$$\dot{\omega}_i(0) = \frac{\omega_{i+1} - \omega_i}{T} + \frac{6\epsilon}{T^2},$$
$$\dot{\omega}_i(T) = \frac{\omega_{i+1} - \omega_i}{T} - \frac{6\epsilon}{T^2}.$$

where ω_i and ω_{i+1} are measured frequencies at the ith data frame, T is the data frame length and ϵ is a small constant ($|\epsilon| < \pi$) introduced in the phase unwrapping procedure to make the difference between wrapped and unwrapped phase an integer multiple of 2π. Note that the second term in $\dot{\omega}_i(0)$ is always equal in magnitude but opposite in sign to the second term in $\dot{\omega}_i(T)$. Thus when no significant frequency change occurs across the frame (*i.e.*, the first term is small), the frequency derivatives at the adjacent two frame boundaries will always be of opposite sign, forcing the frequency track within each frame to have a (either normal or upside-down) bowl-shape. In general, these 'side lobes' will ride on top of the average frequency slope $\frac{\omega_{i+1}-\omega_i}{T}$ (unless $\epsilon = 0$, in which case the phase is quadratic). When the frequency slope is large, one normally would not see those small ripples on top of the large frequency variation due to diminished contribution of the second term. Although the oscillation is typically small and hardly noticeable, and may be negligible for synthesis of speech, it is deemed undesired for synthesis of musical tones.

A third advantage of the proposed algorithm is that it reduces the parameter storage requirement and the computational complexity. After completing the least square fitting, we can choose to store the fitted frequency samples at the frame boundaries in place of the measured ones. Since the quadratic phase model has only three coefficients, they are completely determined by the fitted frequency samples at the frame boundaries and the phase continuity condition. This eliminates the need to store the phase samples at the frame boundaries and simplifies the computation needed to determine the polynomial coefficients. The fitted phase track can be obtained by integration of the instantaneous frequency, which is linearly interpolated from the fitted frequency samples. Although this simple scheme will allow accumulation of round-off errors across the data frame, some alternative schemes given in the next section will achieve the storage saving without incurring the error accumulation.

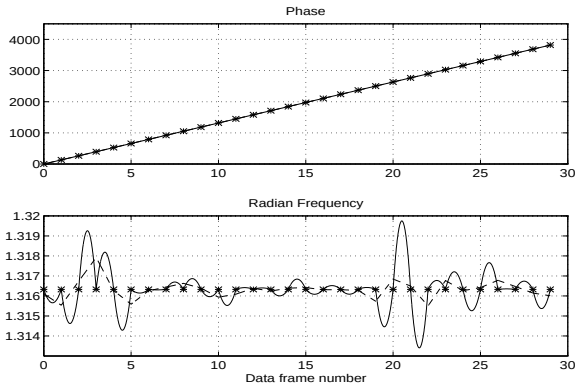

Figure 1: Interpolated frequency tracks obtained from the cubic interpolation algorithm (solid line) and the proposed quadratic interpolation algorithm (dotted line) for a special case where the frequency (stars) is constant and the phase measurements at frame boundaries contain 1% random perturbation.

The block diagram in Figure 2 shows how our least square interpolation algorithm affects the complete analysis/synthesis algorithm in [1]. Note that the fitted phase samples that need not be saved (or transmitted) are denoted by a dashed line. At the analysis stage, the original cubic interpolation algorithm is replaced by a quadratic one. The increase in computational cost due to this change is not critical for many music synthesis applications where analysis can be done off-line. At the synthesis stage, our algorithm yields savings in both computational time and storage space.

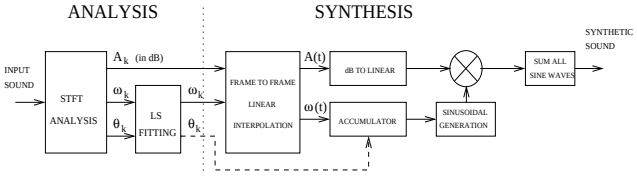

Figure 2: A block diagram of the complete analysis/synthesis system. Note that the fitted phase samples that need not be saved (or transmitted) are denoted by a dashed line.

2 Proposed quadratic phase interpolation algorithm

We model the phase function within each data frame as a quadratic polynomial. Thus the phase and frequency in, say, the ith $(0 \le i < N)$ data frame, can be written as

$$\theta_i(\tau) = a_i + b_i\tau + c_i\tau^2, \qquad \omega_i(\tau) = b_i + 2c_i\tau, \quad (1)$$

where $\tau = t - t_i$ $(0 \le \tau < T)$ and T is the frame length. The polynomial coefficients are determined by minimizing the following error

$$E = \lambda \sum_{i=0}^{N} (\theta(t_i) - \theta_i)^2 + (1-\lambda)T^2 \sum_{i=0}^{N} (\omega(t_i) - \omega_i)^2 \quad (2)$$

subject to the phase and frequency continuity constraints. In the above equation, θ_i and ω_i denote, respectively, phase (unwrapped as in [1]) and frequency at frame boundaries measured from short time Fourier analysis. The polynomial coefficients in Eq. (1) obtained from the minimization can be expressed [4] as

$$\begin{aligned} a_i &= \tfrac{1}{2}(\alpha_{i-1} + \alpha_{i-2}), \\ b_i &= \tfrac{1}{T}(\alpha_{i-1} - \alpha_{i-2}), \\ c_i &= \tfrac{1}{2T^2}(\alpha_i - 2\alpha_{i-1} + \alpha_{i-2}). \end{aligned} \quad (3)$$

where α_i's are solved from the following linear system

$$A\alpha = \lambda(\Theta_0 + \Theta_1) + 2(1-\lambda)T(\Omega_0 - \Omega_1), \quad (4)$$

where A is an $N+2$ by $N+2$ symmetric tridiagonal matrix with the main diagonal $[\frac{a}{2}, a, \cdots, a, \frac{a}{2}]$ and the first diagonal $[b, \cdots, b]$ with

$$\begin{aligned} a &= \lambda + 4(1-\lambda) \\ b &= \frac{\lambda}{2} - 2(1-\lambda). \end{aligned}$$

The other variables in Eq. (4) are given by

$$\begin{aligned} \alpha &= [\alpha_{-2}, \alpha_{-1}, \cdots, \alpha_{N-1}]', \\ \Theta_0 &= [0, \theta_0, \cdots, \theta_N]', \\ \Theta_1 &= [\theta_0, \cdots, \theta_N, 0]', \\ \Omega_0 &= [0, \omega_0, \cdots, \omega_N]', \\ \Omega_1 &= [\omega_0, \cdots, \omega_N, 0]'. \end{aligned}$$

Detailed study is yet to be made to obtain an optimal choice of λ. However, it can be seen that when $\lambda = \frac{4}{5}$, $b = 0$ and the matrix A in Eq. (4) becomes diagonal. It is noticed [3, 4] that in this case, the proposed algorithm can be implemented in real time at either analysis or synthesis stage without causing more than one frame delay or requiring to store the inverse of the system matrix A in (4). In other words, for this case the 'LS FITTING' block in Figure 2 can be moved to the synthesis side for real-time synthesis.

We do *not* recommend using $\lambda = 0$ or $\lambda = 1$ although exact fitting can be achieved in these two cases. As can be seen from Eq. (2), these two λ's correspond to the cases where phase and frequency estimates are, respectively, ignored. From our test results, the quality of the synthesized sounds resulting from these two extreme cases is not of satisfactory.

As diagrammed in Figure 2, the linear system (Eq. (4)) is normally solved at the analysis stage. At the synthesis stage, a phase sample can be computed using two add operations as follows. Note that the finite difference of a quadratic polynomial is a linear function. Thus from Eq. (1), we have

$$\bar{\omega}_i(n) \equiv \theta_i(n+1) - \theta_i(n) = (b_i + c_i) + 2nc_i,$$

assuming the sampling interval is one. A phase sample can then be computed with the following recursion:

$$\begin{aligned} \theta_i(n+1) &= \theta_i(n) + \bar{\omega}_i(n), \\ \bar{\omega}_i(n+1) &= \bar{\omega}_i(n) + (2c_i), \end{aligned}$$

where $0 < n < T$ and the recursion can be initialized by $\theta_i(0) = a_i$ and $\bar{\omega}_i(0) = b_i + c_i$. The coefficients a_i, b_i, and c_i can be obtained in at least three ways. They can be computed from stored α_i coefficients using Eq. (3). Alternatively, we can set a_i to be the phase value at the end of the preceding frame, store b_i, and compute c_i by

$$c_i = \frac{b_{i+1} - b_i}{2T}. \quad (5)$$

The third way is to store both a_i and b_i, and compute c_i using Eq. (5). The first and the second choices need only to store one parameter per frame. The first and the third choices do not incur accumulation of round-off error across the frame boundaries, which can occur in the second choice as a result of using the phase value at the end of the preceding frame as the initial phase of the current frame. Thus those two might be good choices when low precision computation is used. The second and the third choices store physically more meaningful parameters (initial phase and frequency values at the frame boundaries), which might be convenient for sound modification. All three methods, however, simplify the computation used in the cubic algorithm to determine the polynomial coefficients.

3 Test Examples

Some preliminary tests of the algorithm will now be presented. The test results presented below are obtained with $\lambda = \frac{4}{5}$ for computational simplicity. Figure 1 shows the

frequency track (dashed line) resulting from the proposed algorithm for the special case where frequency is constant and phase measurements at frame boundaries contain 1% random perturbations. It can be seen that although the fitted frequencies deviate from the measured ones at frame boundaries, the overall track is closer to the true one and is smoother than the track (solid line) obtained from the cubic interpolation algorithm. In Figure 3, we compare the proposed algorithm with the cubic algorithm for a trumpet note. The top panel shows the frequency tracks of the lowest fifteen harmonics (or partials). There one can barely see any difference of the two algorithms because of the large frequency range plotted. The bottom panel zooms in a portion of the first harmonic. Here the difference between the two algorithms becomes apparent and is similar to what is shown in Figure 1. For this example, we found that the perceived sound quality of the synthesized trumpet notes obtained from the cubic and quadratic interpolation algorithms is not distinguishable.

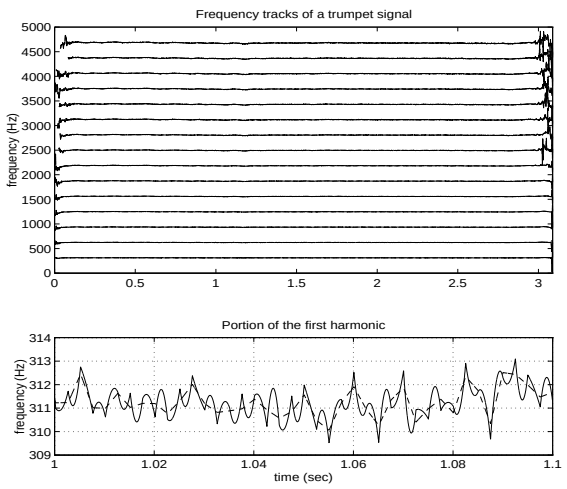

Figure 3: Interpolated frequency tracks obtained from the cubic interpolation algorithm (solid line) and the proposed quadratic interpolation algorithm (dashed line) for a trumpet note.

4 Conclusions

We have described a least square quadratic phase interpolation algorithm, which improves the analysis and synthesis algorithm proposed in [1]. The algorithm uses two add operations to generate each phase sample and needs to store one parameter per data frame for each phase track. Compared with the cubic phase interpolation algorithm in [1], the proposed one is more efficient in terms of computational complexity and parameter storage. In the meantime, it produces smoother frequency tracks (that is, with less spurious oscillations). Unlike the existing quadratic phase interpolation algorithm, where the phase measurements are totally ignored, the proposed algorithm interpolates phase in a least square sense from both the phase and the frequency measurements at frame boundaries. Thus the resulting phase samples are approximately locked to the measured ones. Informal listening tests on various musical instrument tones indicate that the proposed algorithm clearly outperforms the magnitude-only synthesis approach and is qualitatively comparable to the cubic one.

Acknowledgment

The authors would like to express their thanks to Dr. Don Shaver of Texas Instruments for his support throughout this project. Dr. Xiaoshu Qian's work was sponsored by Texas Instruments Incorporated through its student internship program.

References

[1] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 744–754, August 1986.

[2] X. Serra and J. O. Smith, "Spectral modeling system: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, 1990.

[3] Y. Ding and X. Qian, "Processing of musical tones using a combined quadratic polynomial-phase sinusoids and residual signal model," *submitted to Journal of Audio Engineering Society*, Dec. 1996.

[4] X. Qian, *Track frequency components in speech and music signals.* PhD thesis, University of Rhode Island, Kingston, RI, 1997.