Optimized Sinusoid Synthesis via Inverse Truncated Fourier Transform

Rade Kutil

Published version:

R. Kutil, "Optimized sinusoid synthesis via inverse truncated fourier transform," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, no. 2, pp. 221-230, Feb. 2009, doi:10.1109/TASL.2008.2004292

Optimized Sinusoid Synthesis via Inverse Truncated Fourier Transform

Rade Kutil

***Abstract*—It was shown that sinusoid synthesis can be implemented efficiently by an inverse Fourier transform on consecutive frames where all but a small number of coefficients per oscillator are dropped. This leads to a compromise between computational complexity and approximation accuracy. The method can be improved by two approaches. First, optimal coefficients can be found by minimizing the average approximation error. Second, the optimal window function can be found through an iterative process. The gain in signal-to-noise ratio (SNR) is between 10 and 40 dB and can be used to reduce computational complexity while satisfying required synthesis quality.**

I. Introduction

In sound synthesis based on spectral modelling [1], [2], [3], [4], large arrays of oscillators must be implemented. Each oscillator generates a signal

$$x(t) = A\sin(2\pi ft + \varphi)\,, \tag{1}$$

where A is the amplitude, f is the frequency and φ the phase. The final signal is the sum of all oscillator's output signals.

A time-domain method to generate sinusoids as in (1) is by means of the finite difference scheme (digital resonator)

$$x(t+1) = 2\cos(2\pi f)x(t) - x(t-1)\,. \tag{2}$$

Thus, each oscillator only requires one multiplication and one subtraction per sample point. Nevertheless, high sampling rates and large numbers of oscillators often impose prohibitive computational demands, although the number of oscillators can be reduced due to psychoacoustics [5]. Moreover, there are problems with numerical stability [6], [7] because (2) represents an IIR filter with two poles on the unit circle at $e^{\pm i2\pi f}$.

To reduce the overall complexity, several methods have been developed to make complexity less dependent on the number of oscillators. A recent approach uses a polynomial generator [8]. More conventional approaches use the inverse Fourier transform, which is still popular in recent applications [9], [10]. The idea is to split the signal into frames of fixed length N which are modelled in the Fourier transform domain. For each oscillator only a few frequency bins should have to be set, since the oscillator operates at a single frequency.

More precisely, the synthesized signal is represented by Fourier coefficients $Y(k)$, where $k = M_0, \ldots, M_1$. Of course, mirrored coefficients $\bar{Y}(-k)$ must also be accounted for. The used bandwidth $M = M_1 - M_0 + 1$ should be as small as possible to minimize computational demands. It cannot be arbitrarily small, though, because aliasing and border effect problems arise when the oscillator frequency does not correspond exactly to a single frequency bin.

The calculation of the $Y(k)$ during synthesis is done by pre-computed coefficients that are relocated in the Fourier domain and modified in amplitude and phase by complex multiplication. This requires a number of operations, i.e. complex multiplications and additions, that is proportional to the bandwidth M. These operations have to be performed once for a whole frame, whereas in the finite-difference implementation, operations are needed for each sample point. Thus, the method amounts to a reduction in complexity by a factor proportional to the frame length.

The coefficients of all oscillators have to be added in the Fourier domain before the signal is generated in the second step, by an inverse Fourier transform

$$y(t) = \sum_{k=M_0}^{M_1} Y(k)e^{ik\frac{2\pi}{N}t}\,, \tag{3}$$

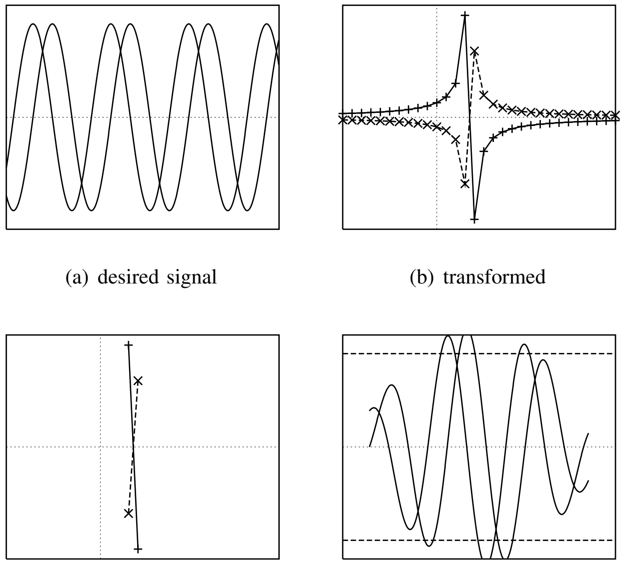
where only the real part of $y(t)$ is used if a real signal is required. The transform can be implemented by an inverse FFT so that the complexity per sample depends logarithmically on the frame size. Thus, the first step decreases as $O(\frac{1}{N})$ and the second step increases as $O(\log N)$ with the frame length N. An optimal compromise has to be found. Moreover, if volume and pitch of the oscillators changes frequently, this may further restrict the frame length because such changes may not be possible during a frame.

Fig. 1 demonstrates the method. Also shown is the calculation of $Y(k)$ by a forward transform of the desired signal. The $Y(k)$ are pre-computed for a sufficiently dense set of frequencies and stored in memory. Fortunately, no more than a range of size 1 is needed for these frequencies because integer steps in frequency can be achieved by a relocation of the coefficients, as is shown later on. Also, phase and amplitude can be controlled easily in the Fourier domain. The pre-computation step can also be viewed as the calculation of an oversampled frequency response through a Fourier transform of the zero-padded frame.

The main problem with this approach is that oscillator frequencies that do not fall exactly onto a single Fourier component occupy the whole spectrum, with magnitudes decreasing only by $O(\frac{1}{d})$ with the distance d from the center frequency. In other words, the rectangular function produced by zero-padding evokes large side-lobes in the frequency response. This means that many frequency bins have to be set in order to satisfy limits on the approximation error.

Most early approaches [11], [12], [13], [14] have therefore

Rade Kutil is with the Department of Computer Sciences, University of Salzburg, Jakob Haringer Str. 2, 5020 Salzburg, Austria, e-mail: rkutil@cosy.sbg.ac.at, phone: +43 662 8044-6303, fax: +43 662 8044-172



(a) desired signal (b) transformed (c) essential coefficients (d) approximated signal

Fig. 1. Approximation of a sinusoid signal with non-integer frequency (3.5) by few ($M = 2$) essential coefficients. Time signals show sine and cosine together to make the shape of the amplitude visible more easily. Spectral plots show the real and imaginary parts of coefficients.

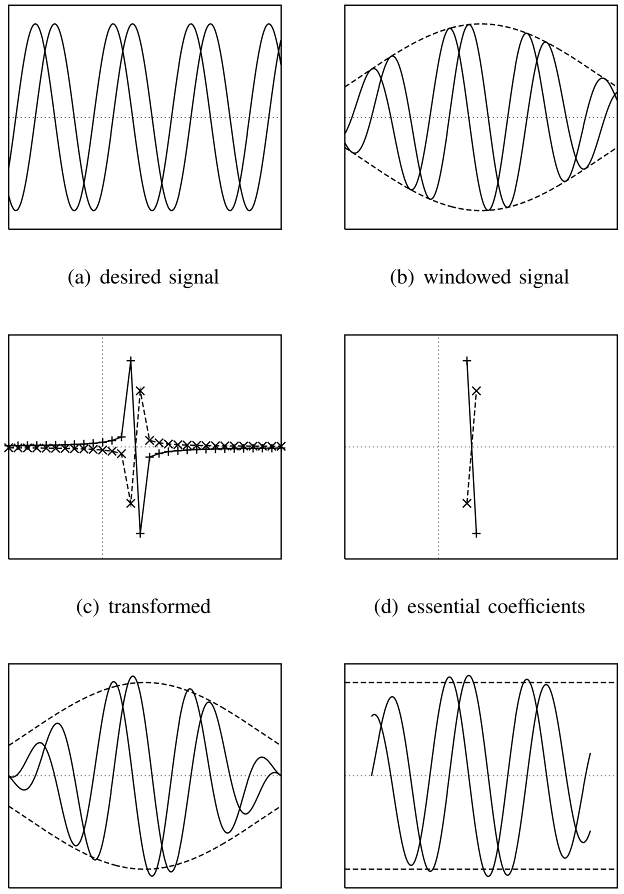
adopted overlap-add techniques and window functions [15], with the aim of minimizing the side-lobes by the choice of the window function. The employed method is basically an inverse short-time Fourier transform. Additionally, the window function can be allowed to be the product of two parts, one that adds up to 1 everywhere, as usual, and one that has to be inverted explicitly after signal generation. This allows for more freedom in the choice of the window function. While all that works well, the computational complexity is increased and problems with changing pitches arise.

In [16], a technique without overlap is developed. For the same reasons, it also uses a window function $h(t)$ which is obviously of the latter kind. Furthermore, the signal is truncated in the time domain by about 10% at the borders, where the approximation error is worst. Thus, the desired signal $x(t)$ is approximated by

$$h(t)x(t) \approx y(t) \qquad \text{for} \qquad t = T_0, \ldots, T_1\,. \tag{4}$$

The values of $y(t)$ for t outside of $[T_0, T_1]$ are dropped. This decreases the computational efficiency somewhat, but since no overlap is required, the method is still favorable. The coefficients $Y(k)$ are found by a Fourier transform of $h(t)x(t)$, those outside of $[M_0, M_1]$ are dropped. After the inverse Fourier transform, the signal is renormalized by multiplication of the inverted window function. There are no problems with changing pitches at the borders. Changing volumes are not so easy to handle, though. Linear changes of volume within a frame are added separately in the frequency domain, causing additional computational effort per oscillator (see Section V).

Fig. 2 demonstrates the method. One can see that the windowed signal is easier to approximate because coefficients outside of the range $[M_0, M_1] = [3, 4]$ are smaller. As a



(a) desired signal (b) windowed signal (c) transformed (d) essential coefficients (e) approximated windowed signal (f) reverse-windowed approximation

Fig. 2. Approximation of a sinusoid signal with non-integer frequency (3.5) by few ($M = 2$) essential coefficients using a Kaiser window to increase frequency concentration.

consequence, the approximation of the desired function by the reverse-windowed function in Fig. 2(f) is better than the direct approximation in Fig. 1(d).

So far, $x(t)$ was real-valued. For theoretical considerations we will now model $x(t)$ as complex-valued. This has two advantages. First, the coefficients in the Fourier domain will not be mirrored in negative frequencies. Second, error integrals over the phase space are turned into taking the squared norm. Once the optimized coefficients are calculated and stored in memory, a real-valued inverse FFT is used where the Fourier domain is assumed to be conjugate mirrored and a real-valued signal is generated. Window functions are still real-valued. Thus, $x(t)$ is modelled as

$$x(t) = x(\alpha, t) = Ae^{i\alpha\frac{2\pi}{N}t} \tag{5}$$

where A is complex and contains the phase and amplitude of the signal, and α is the frequency of the oscillator in relation to the frame length. A has to be set so that subsequent frames fit together, i.e. $A_{m+1} = A_m e^{i\alpha\frac{2\pi}{N}T}$ where $T = T_1 - T_0 + 1$. Since phase and amplitude of synthesized signals can be manipulated easily by multiplying the coefficients $Y(k)$, the rest of this paper will assume $A = 1$ without loss of generality.

α does not have to be integer and has an arbitrary range. It is obvious that the frequency bins that are best suitable to approximate the signal are those closest to α. In other words, $|\frac{1}{2}(M_1 + M_0) - \alpha| \leq \frac{1}{2}$ should be fulfilled. Moreover, if α is increased by an integer w, the Fourier coefficients $Y(k)$ can be reused by shifting them up w frequency bins. Therefore, we only need to consider the cases $-\frac{1}{2} \leq \alpha \leq \frac{1}{2}$ for uneven numbers M of frequency bins, and $0 \leq \alpha \leq 1$ for even M.

The method of truncating the output of the inverse Fourier transform raises the question whether the chosen Fourier coefficient values and window functions are optimal in terms of approximation SNR. The problem of finding minimal-bandwidth representations of incompletely defined functions is generally known as inter- and extrapolation problem [17]. In the band-limited case, the well-known Papoulis-Gerchberg algorithm [18] solves the problem by a recursion of alternating band-limiting and resubstitution of the desired signal in non-truncated parts of the frame. It converges to the solution with the minimal approximation error

$$\sum_{t=T_0}^{T_1} |g(t)y(t) - x(t)|^2 \qquad \text{where} \qquad g(t) = \frac{1}{h(t)}\,. \tag{6}$$

This solution can also be found through systems of linear equations, though. One such approach, based on Fourier coefficients, is developed in Section II.

The second question is how to find the optimal window function $h(t)$ (or $g(t)$). For this purpose, the approximation error (6) for $x(\alpha, t)$ is averaged over the whole range of α. Then, it is minimized by the choice of the window function. The resulting equation is solved in Section III through an iterative method that converges quite slowly, but improves the approximation accuracy significantly over the Kaiser or Tchebychef windows used in [16]. As a result, fewer frequency bins have to be used and the frame does not have to be truncated as much.

II. Optimal Coefficients

In [16] the window function $h(t)$ is defined on the non-truncated frame $[0, N-1]$. The Fourier coefficients $Y(k)$ are calculated by a forward Fourier transform of the windowed signal $x(t)h(t)$, where $t = 0, \dots, N-1$. Although coefficients that are found in this way are good, they are not optimal.

Therefore, we will try to find Fourier coefficients $Y(k)$ that best approximate the signal $x(t)$. The range of frequency bins $[M_0, M_1]$ and the window function $h(t)$ is fixed. The approach is to minimize the approximation error (6), which is a function of the Fourier coefficients $Y(k)$ from (3). To find the optimal coefficients, we simply differentiate (6) with respect to $Y(n)$ for $n = M_0, \dots, M_1$ and set the result to zero. We get

$$\sum_{k=M_0}^{M_1} Y(k) \sum_{t=T_0}^{T_1} g(t) e^{i(k-n)\frac{2\pi}{N}t} = \sum_{t=T_0}^{T_1} x(t) e^{-in\frac{2\pi}{N}t}\,. \tag{7}$$

See Appendix A for a derivation of this result. There are two appearances of a “truncated Fourier transform” in this

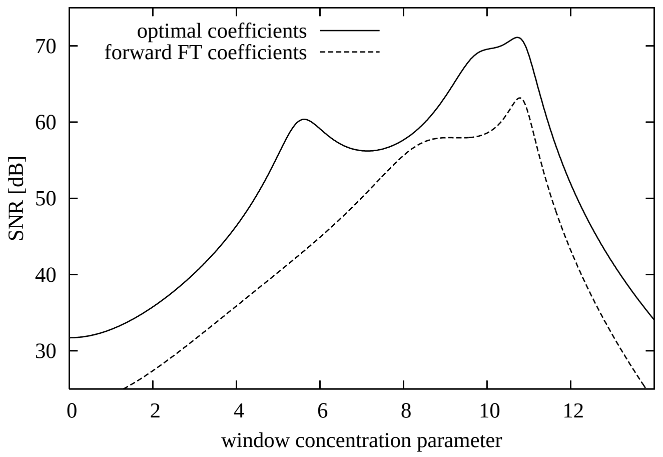


Fig. 3. Approximation quality for the Kaiser window, comparing the conventional forward FT method to optimal coefficients. $M = 7$, $N = 1024$, $T = 824$. The SNR in dB is shown depending on the concentration parameter of the Kaiser window.

equation. If we define this transform as $f \mapsto \tilde{F}$,

$$\tilde{F}(n) := \sum_{t=T_0}^{T_1} f(t) e^{-in\frac{2\pi}{N}t}\,, \tag{8}$$

then we can reformulate (7) as

$$\sum_{k=M_0}^{M_1} Y(k) \tilde{G}(k-n) = \tilde{X}(n)\,. \tag{9}$$

This is basically a system of linear equations of size $M \times M$, where $\tilde{G}$ forms a Toeplitz matrix. The solution is unique and minimizes the approximation error.

Fig. 3 shows how the optimal coefficients improve the accuracy over the forward FT method in [16]. For $M = 7$, $N = 1024$, $T = 824$ and $\alpha = -0.5, \dots, 0.5$ (in short steps) the coefficients and the signal-to-noise ratio (SNR) between y and x are calculated with either method. The gain is about 8 dB between the optima, where the optima are almost but not exactly located at the same concentration parameter of the Kaiser window.

III. Optimal Window Function

The window function must not depend on the frequency α. This is an important concept to allow the second step in the synthesis, i.e. the inverse Fourier transform together with application of the inverted window, to be executed once for all oscillators whose frequency components are added in the transform domain. Therefore, the window function has to be chosen so that the sum of approximation errors over the relevant range of α is minimal. Thus, we have to minimize

$$\int_{\alpha_0}^{\alpha_1} \sum_{t=T_0}^{T_1} |g(t)y(\alpha, t) - x(\alpha, t)|^2 d\alpha \tag{10}$$

by varying the values $g(t)$ of the window function. Again, we do this by differentiating (10) with respect to $g(t)$ and setting the result to zero. In this way, we arrive at the following set of equations:

$$g(t) \int_{\alpha_0}^{\alpha_1} |y(\alpha,t)|^2 d\alpha = \int_{\alpha_0}^{\alpha_1} x(\alpha,t)\overline{y(\alpha,t)} d\alpha\,. \tag{11}$$

See Appendix B for a derivation of this result.

Altogether, the error function (10) can be viewed as a function of the union of all $Y(k)$ and $g(t)$. If the $g(t)$ are fixed, then the error function has a unique minimum due to (9). On the other hand, if the $Y(k)$ are fixed, then it has also a unique minimum because (11) has only one solution. Now, if a solution can be found that satisfies both conditions, does this constitute a global minimum? This would certainly be the case if the Hessian matrix were positive everywhere or, equivalently, if the second directional derivatives D_v^2 along all vectors v are positive, where v is a vector in the space of $Y(k)$ and $g(t)$. These derivatives are

$$\int_{\alpha_0}^{\alpha_1} \sum_{t=T_0}^{T_1} 2(D_v g(t) y(\alpha,t))^2 + 2(g(t)y(\alpha,t) - x(\alpha,t)) D_v^2 g(t) y(\alpha,t)\, d\alpha\,. \tag{12}$$

The first part of this expression is positive except for degenerate cases, if ever. The second part incorporates the approximation difference as a factor. Therefore, it is small compared to the first part if the approximation is not entirely bad. Thus, this second directional derivative of the error function is positive for reasonable regions of the parameter space. As a consequence, there is a unique minimum that can be found by solving (9) and (11) simultaneously, unless there is a very different optimal solution, which is unlikely.

The two equations cannot be solved together directly because the involved expressions depend on Y and thus in turn on g in an entirely non-linear way. However, it is possible to develop an iterative method. By alternatingly calculating the optimal Y from g by (9) and g from Y by (11), we get a scheme

$$g_j(t) \longrightarrow \tilde{G}_j(k) \xrightarrow{(9)} Y_j(\alpha,k) \longrightarrow y_j(\alpha,t) \xrightarrow{(11)} g_{j+1}(t) \tag{13}$$

that converges to the optimal solution, since convergence implies both conditions (9) and (11). Note that the coefficients and the window function are optimized simultaneously since each iteration involves the optimization of both.

The convergence is rather slow, the number of necessary iterations seems to depend exponentially on the number M of frequency bins. Fig. 4 shows how the SNR of the approximation approaches the maximum as the number of iterations grows. The convergence happens in about $\frac{M-1}{2}$ "waves" with approximately equal widths on the logarithmic scale. Thus, for high bandwidths M the calculation times for the optimal window function can be ridiculously long. Fortunately, small M are sufficient and desirable in our case. Moreover, the optimal window and coefficients only have to be obtained once, prior to the real-time synthesis.

Note also that the window function could be complex in principle. However, the imaginary part always comes out as zero because $x(\alpha,t)$ and $y(\alpha,t)$ have equal phases, so the term $x(\alpha,t)\overline{y(\alpha,t)}$ in (11) has zero phase.

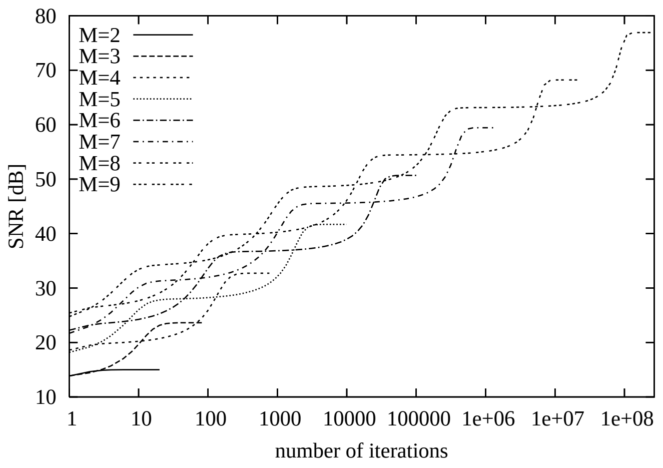


Fig. 4. Convergence of the iterative method to find the optimal window function. The SNR in dB of the approximation is shown depending on the number of iterations. $N = 1024, T = 960, M = 2,\ldots,9$.

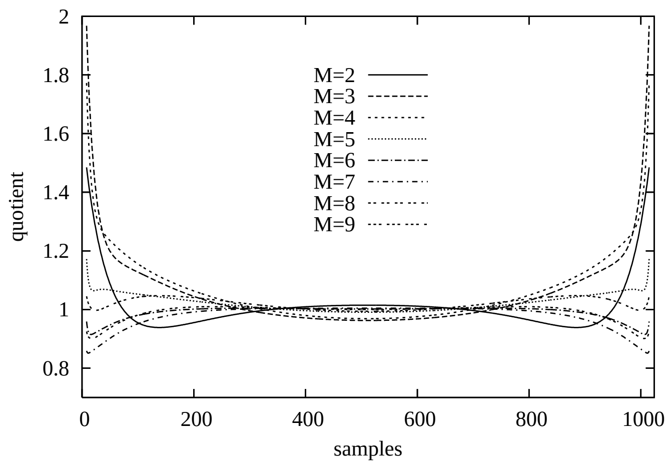
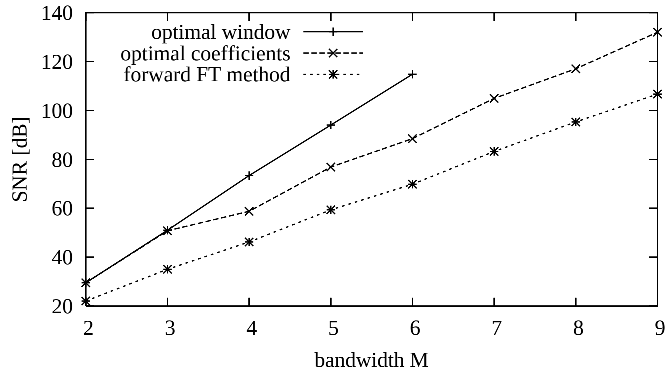


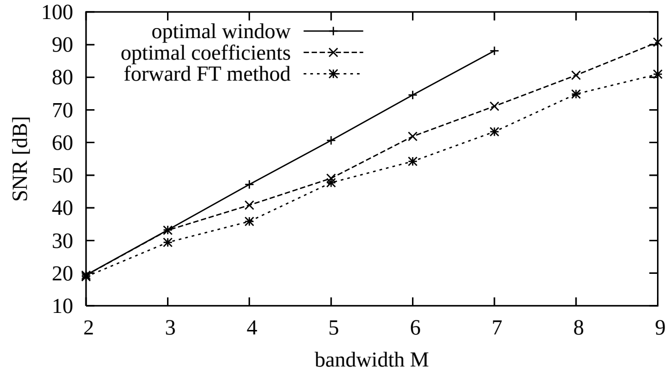
Fig. 5. Quotient of the optimal window and the best Kaiser window for $N = 1024$, $T = 1008$ and $M = 2,\ldots,9$.

The optimal window is very similar in shape and scale to the Kaiser window. Fig. 5 shows the quotient of the optimal window and the best Kaiser window for $N = 1024$, $T = 1008$ and $M = 2,\ldots,9$. The two windows are scaled so that they have both the same integral between T_0 and T_1 before building the quotient. One can see that the main difference lies at the borders of the frame, where the approximation is most difficult.

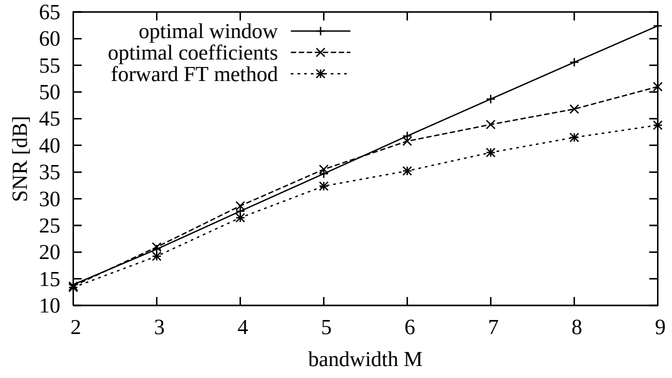
A comparison of SNR results is shown in Fig. 6. Three methods are compared. The first one uses the optimal Kaiser window, where the Fourier coefficients are calculated with the forward FFT method as in [16], the second one uses the optimal Kaiser window and optimal Fourier coefficients as in (9), and the third one uses the optimal window developed in this section. In the interesting range from 40 to 80 dB, the optimal window gains 10 to 40 dB compared to the forward FFT method, where 5 to 20 dB are due to optimal coefficients. The fact that the optimal window seems to perform slightly



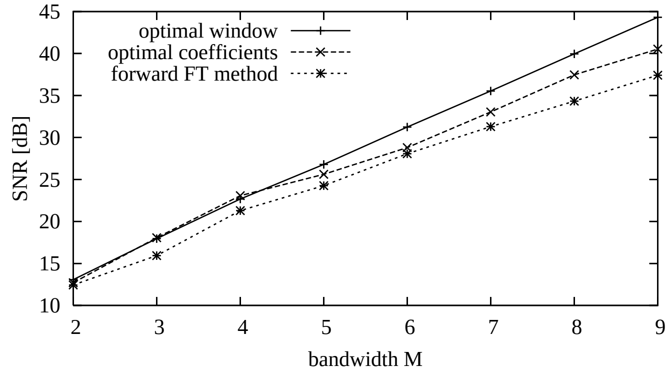
(a) $T = 512$



(b) $T = 824$



(c) $T = 992$



(d) $T = 1016$

Fig. 6. SNR of the three methods for $N = 1024$, various values of T and $M = 2, \ldots, 9$.

worse than the Kaiser window with optimal coefficients for a few points is most likely caused by numerical imprecision. The improvement is greater for smaller T, which means that the Kaiser window is less fit for heavily truncated signals.

IV. Varying Pitch

So far, only oscillators with constant pitch and volume have been considered. A change in pitch can be realized easily at the frame borders. The only thing to ensure is that phases fit together at the transition from one frame to the next. Sudden changes in pitch are widely believed to not impose unpleasant artifacts if the changes are small enough [19].

In IFFT methods the intervals where the frequency of an oscillator can be changed are determined by the frame size. This raises the question how large these intervals can be without producing an audible distortion. Because the author could not find an appropriate investigation of this issue in literature, a small ad hoc experiment might help.

In this experiment, a test person is presented with two chirp signals with a duration of one second, spanning a certain frequency range from f_L to f_H. One of the signals has a perfect continuous exponential frequency ascent, the other one is stepped in intervals of a certain step length. The two signals are ordered randomly and the test person has to guess which one is the stepped chirp.

Depending on how well the test person can distinguish the signals, there is a probability p of a correct guess. If the signals are easily distinguishable, then $p = 1$, if they are indistinguishable, then $p = 0.5$. The process is repeated until it can be concluded that p is significantly above or below 0.75. This is the case if the number k of correct out of n total guesses yields a probability $P(l \geq k \,|\, p = 0.75)$ or $P(l \leq k \,|\, p = 0.75)$ of at most 20%, respectively, where l is binomially distributed, $l \sim B_{n,p}$.

The critical step length is found by a binary search, based on the above decisions. This is done for the whole range of frequencies and frequency gradients. The author admits that the experiment does not constitute a thorough psychoacoustical investigation for several reasons, such as cheap audio equipment and enlisting only a single test person who is identical to the author. However, the procedure guarantees small deviation and the aim is only a feasibility study.

Fig. 7 shows the results. The mean frequency on the horizontal axis is calculated in the logarithmic sense because the chirps are of the exponential type, i.e. $\bar{f} = \sqrt{f_L \cdot f_H}$. The gradients of the chirps are accordingly measured in cent per second. One can see that a stepped frequency glide can be detected more easily for high gradients and high frequencies. Reasonable step lengths are between 5 and 50 ms, which easily allows for frame sizes that are beneficial for our IFFT method.

V. Varying Volume

To cope with volume changes, requires additional effort. The finite difference scheme principally requires two multiplications and mostly also an increased number of additions per sample point [20]. In [19] the overhead is reduced to a small percentage by changing oscillator parameters only in

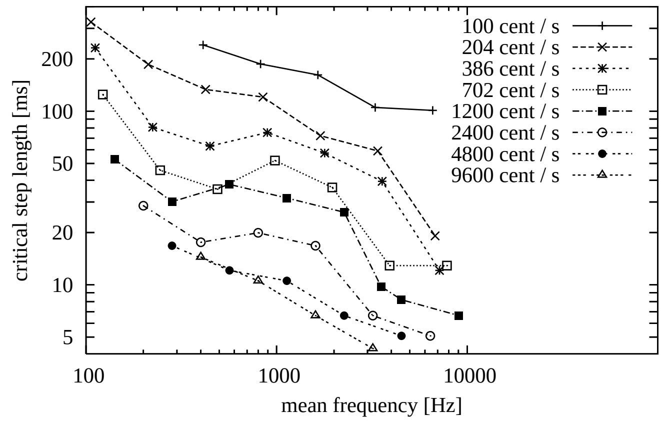


Fig. 7. If the step length of a stepped frequency glide is greater than the critical step length, it can be distinguished from a perfect glide.

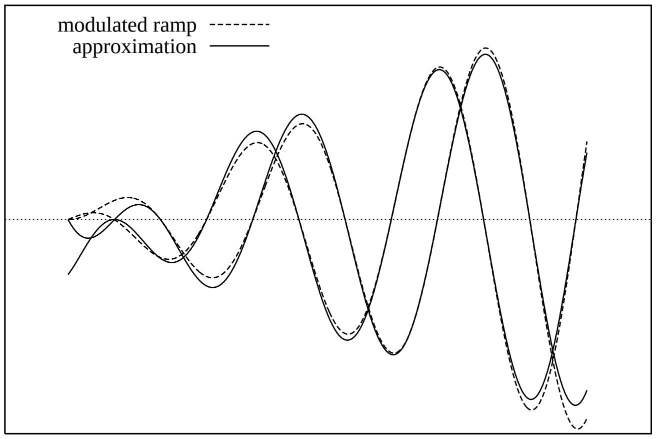


Fig. 8. Approximation of a modulated ramp signal for frequency 3.5 (relative to FFT-size), $T = 824$ and $M = 2$.

certain intervals, at points which are claimed to produce least audible artifacts, such as maxima and zero-crossings.

For the IFFT scheme, [16] suggests to create an extra signal with a frequency equal to that of the oscillator but a volume that increases linearly from 0 to a certain amount. This signal is also modelled in the Fourier domain and added to the constant-volume oscillator coefficients. By scaling this signal, the rate of volume change can be chosen arbitrarily.

To include such a signal into our framework, we simply have to model $x(t)$ in (5) as modulated ramp signal

$$x(t) = A(t - T_0)e^{i\alpha \frac{2\pi}{N} t} \tag{14}$$

and proceed as in Section II. The window function used for calculating the optimal volume-change coefficients is the optimal window calculated for the constant-volume signal. This has the advantage that the synthesis coefficients can be added to the constant-volume ones and there is no need for an extra FFT. The disadvantage is somewhat suboptimal accuracy. However, accuracy is less crucial when the volume is not constant.

Fig. 8 shows an example for a small $M = 2$ to make the approximation error visible.

VI. Complexity and Best Parameter Choice

The reason to implement oscillator arrays through an inverse FFT is to reduce overall computational complexity. This is achieved by a shift of complexity per oscillator into complexity per sample point. The average number of multiply m and add operations a per sample point depend on several parameters: the size N of the FFT, the frame size T, the number of frequency bins M per oscillators, and the number O of oscillators. These parameters also determine the average SNR of the signal approximation.

Increasing N decreases m and a up to a certain point after which they increase again. Increasing T decreases m and a but also decreases the SNR. For M the situation is reversed. High numbers of oscillators O obviously imply high numbers of operations m and a. Additionally, O influences how other parameters must be chosen to minimize the complexity.

Usually, O and a minimal SNR are given. The other three parameters N, T, M should be optimized for minimal m and a. However, this approach quickly leads to unrealistically high values of T, which is undesirable because oscillators cannot be altered during T samples. Therefore, we will also consider T as given and fixed and find optimal values of N and M.

To do this, we first have to determine the number of multiplies m_i and adds a_i required in each of the two steps, i.e. in the calculation of the Fourier coefficients (m_1, a_1) and in the inverse FFT (m_2, a_2). The former requires one complex multiplication to determine the phase and amplitude of the oscillator in the frame, i.e. A in (5), plus one complex multiplication of A with each coefficient $Y(k)$. The results have to be added to the Fourier domain representation of the frame. As each complex multiplication requires 4 real multiplications and 2 additions, we get

$$Tm_1 = 4OM + 4O, \qquad Ta_1 = 4OM + 2O\,. \tag{15}$$

The FFT part can be implemented by an inverse FFT with positive frequency input only and real output. Optimization thereof leads to the following numbers of operations:

$$\begin{aligned} Tm_2 &= 2N\log_2 N - 9N + 6\log_2 N + 8\,, && (16)\\ Ta_2 &= 3N\log_2 N - 8N + 12\,, && (17) \end{aligned}$$

where N is a power of 2. It turns out that, for higher numbers of oscillators O, the optimal T is significantly smaller than N. In this case, the complexity of the inverse FFT can be further reduced by a truncated FFT [21] which cuts off all operations that are not needed for the requested output. See Fig. 9. More precisely, Tm_2 is reduced by

$$\begin{aligned} &\sum_{j=T}^{N-1} \left(2^{\lceil l(j)\rceil+1} - 2\lceil l(j)\rceil - 2\right)\\ &\quad - \chi(l(T) \geq 2)\left(9 \cdot 2^{\lfloor l(T)\rfloor - 1} - 4\lfloor l(T)\rfloor - 4\right)\\ &\quad + \chi\left(l(T) \geq 1 \wedge T > \frac{3}{4}2^{n-\lfloor l(T)\rfloor}\right)\left(2^{\lfloor l(T)\rfloor} - 1\right)\\ &\quad - \chi(\lfloor l(T)\rfloor = 1)\,, \end{aligned} \tag{18}$$

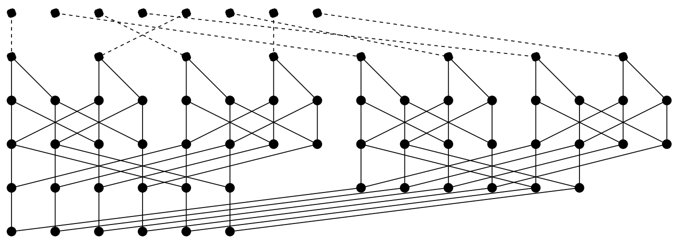


Fig. 9. Truncated inverse FFT scheme for $N = 16$ and $T = 6$. The input in the first row consists of positive frequency coefficients which are sorted in the second row in the usual bit-reverse order. The last row is the output signal of length T.

where $l(t) = \log_2(N) - \log_2(t)$, and $\chi(A) = 1$ if A is true, and $\chi(A) = 0$ else. Accordingly, Ta_2 is reduced by

$$\sum_{j=T}^{N-1} \left(3 \cdot 2^{\lceil l(j) \rceil} - 2\lceil l(j) \rceil - 3\right) + \chi(l(T) \geq 2) \left(2^{\lfloor l(T) \rfloor} - 2\lfloor l(T) \rfloor\right) . \quad (19)$$

Formulas (16)–(19) have been found by the author by analysing empirical counts of an implementation of the truncated FFT and are exact unless N is very small.

After the two steps, the result has to be multiplied by the inverted window function $g(t)$, which adds another real multiplication per sample point. Together, $m = m_1 + m_2 + 1$ and $a = a_1 + a_2$ represent the average number of multiplies and adds per sample point.

Now, we notice that the SNR depends, apart from M, only on the fraction T/N. This follows from the fact that, if N and T are increased by a factor, then synthesized sinusoids for a certain range of α are just dilated by the same factor if the range of α and the coefficients are retained. As a consequence, the coefficients are still optimal and the average squared error remains the same.

Taking this as a basis, we calculate optimal windows and coefficients for a range of values of M and T while N is fixed, say 1024. Then, for a larger range of N, M, and O, and some representative values of T, we calculate approximated optimal SNRs through interpolation, as well as the average number of multiply and add operations per sample point. For some values of minimal SNR, we remember those parameter combinations that minimize the number of operations, i.e. $2m + a$ to be precise. The choice of how to weight m and a is somehow arbitrary, and for modern architectures $m + a$ seems more appropriate. However, m is a better estimator for iteration counts and, thus, supposed to incorporate loop control overhead.

The result is shown in Table I. The table is actually three-dimensional, as it depends on T, SNR, and O. However, the optimal parameters N and M are constant over wide ranges of O. Therefore, the O-dimension is pruned to the lowest O of each range. Thus, for values O' not in the table, the optimal parameter choice is that of the largest O smaller than O', whereupon m is increased by $(4M + 4)(O' - O)/T$ and a by $(4M + 2)(O' - O)/T$. The maximal considered O is 50,000.

The table suggests that there are lower bounds on O and T for the method to be superior to the finite difference scheme, where $m = O$ and $a = 2O$. Although these bounds depend on the SNR and on each other, a rule of thumb may be $O > 20$ and $T > 50$. The performance gain improves dramatically for larger T and O. The ratio T/N is close to 1 for small O, but decreases for larger O because this permits smaller values of M while retaining the approximation quality. Thus, complexity per oscillator is shifted to complexity per sample point which is less significant for larger O. For the same reason, the ratio T/N also decreases with increasing SNR.

Note that, if volume changes are necessary and implemented as in Section V, then the first step requires twice as many operations, i.e. m_1 and a_1 are doubled. This can be accounted for by letting O be twice the number of oscillators when looking up optimal parameters in Table I.

Audio examples are provided at [22].

VII. Practical Performance

Section VI presents complexity as the theoretical performance of the algorithm. This performance can be compared directly with that of [16] since the resulting program code to generate oscillatory signals is identical. Only the used coefficients and window functions differ. If the minimum SNR is met with smaller bandwidth by using the optimized parameters (see Fig. 6), then the performance increases accordingly. Note also that, as claimed in [16], the non-overlapping technique is as fast as, or at most 25% slower than overlap-add techniques, but delivers signals of higher quality, so it is, in the end, faster with respect to equal quality.

Nevertheless, it is interesting to compare the method to fast finite difference methods. The question is whether compiled code contains more overhead per effective multiplication or addition for one method than for the other. Therefore, an implementation of the fast technique in [6] is investigated. It applies the usual scheme from (2) on every fourth sample and calculates the remaining samples by appropriate nested interpolation. While this leaves the number of multiplications and additions unchanged, some instructions become independent and can be scheduled to avoid processor stalls. Additionally, loop unrolling can further improve the performance. While [6] suggests unrolling 3 iterations, the author's experiments show that unrolling 8 iterations performs best.

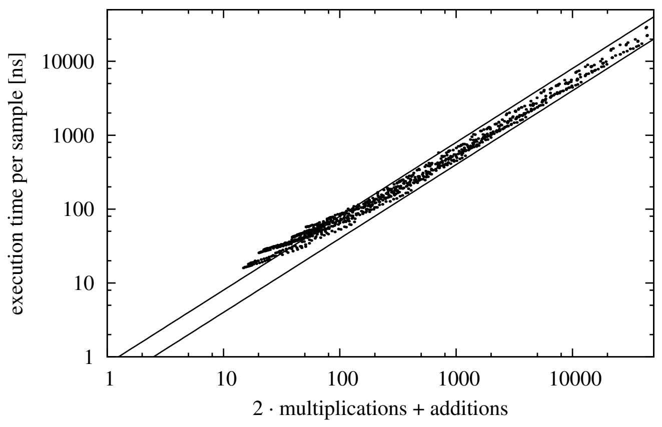
Both algorithms have been implemented with an equal amount of optimization, i.e. pointers to access arrays, single precision floating-point data and pre-computed constants. gcc 4.1.2 has been used with option –O3 on an Intel Pentium 4 CPU with 3.2GHz. SSE operations have not been used. The truncated FFT is implemented by automatically generated flat C-code without loops.

The execution times of each method can now be compared to their theoretical complexity, i.e. $c = 2m + a$, where m is the number of effective multiplications per sample and a the number of additions. The complexity of the IFFT method is described in Section VI. The complexity of the finite difference scheme includes one multiplication per sample ($m = 1$), one addition as shown in (2), plus one addition due to the summation of the oscillators ($a = 2$). Thus, $c = O \cdot (2 \cdot 1 + 2) = 4O$, where O is the number of oscillators.

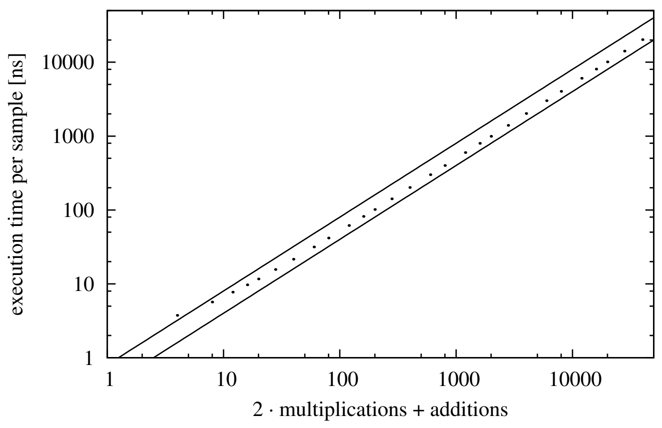
TABLE I

OPTIMAL SYNTHESIS PARAMETERS DEPENDING ON THE FRAME SIZE T, THE NUMBER OF SYNTHESIZED OSCILLATORS O AND THE MINIMUM APPROXIMATION QUALITY (SNR). EACH ENTRY ALSO CONTAINS THE FFT SIZE N, THE BANDWIDTH M, THE ACTUAL SNR, AND THE NUMBER OF MULTIPLIES m AND ADDS a PER SAMPLE POINT. EACH ENTRY MINIMIZES $2m + a$ BY THE CHOICE OF N AND M. FOR VALUES O' NOT IN THE TABLE, THE OPTIMAL PARAMETER CHOICE IS THAT OF THE LARGEST O SMALLER THAN O', WHEREUPON m IS INCREASED BY $(4M + 4)(O' - O)$ AND a BY $(4M + 2)(O' - O)$.

min. SNR	30			40			50			60			70			80		
	O	N	M	O	N	M	O	N	M	O	N	M	O	N	M	O	N	M
T	m	a	SNR	m	a	SNR	m	a	SNR	m	a	SNR	m	a	SNR	m	a	SNR
10	1	16	3	1	16	3	1	16	4	1	16	4	1	16	5	1	16	5
	4.2	9.2	43.7	4.2	9.2	43.7	4.6	9.6	62.6	4.6	9.6	62.6	5	10	80.4	5	10	80.4
	22	32	2	76	64	2	22	32	3	22	32	3	22	32	4	22	32	4
	34.5	45.1	39	113.6	134.6	51.3	43.3	53.9	65.9	43.3	53.9	65.9	52.1	62.7	94.6	52.1	62.7	94.6
							54	64	2	176	128	2	54	64	3	54	64	3
							87.2	112.6	51.3	267.4	313.9	63.4	108.8	134.2	84.7	108.8	134.2	84.7
													400	256	2	1022	512	2
													614.2	713.8	75.4	1540	1725	83
50	1	64	3	1	64	4	1	64	5	1	64	5	1	64	6	1	64	7
	6.04	14.08	34.6	6.12	14.16	49.2	6.2	14.24	63.2	6.2	14.24	63.2	6.28	14.32	77.7	6.36	14.4	93.1
	185	128	2	185	128	3	93	128	3	185	128	4	93	128	4	62	128	4
	60.64	73.98	35.1	75.44	88.78	59.7	46	63.02	59.7	90.24	103.6	85.8	53.44	70.46	85.8	41.04	59.3	85.8
				451	256	2	1497	512	2	451	256	3	451	256	3	1497	512	3
				151.4	181.6	47.5	466.2	514.1	59.6	187.5	217.7	78.9	187.5	217.7	78.9	586	633.8	97.3
										3408	1024	2	3408	1024	2	7620	2048	2
										1071	1170	71.7	1071	1170	71.7	2411	2616	81.1
100	1	128	3	1	128	4	1	128	5	1	128	5	1	128	6	1	128	7
	8.06	17.76	34.6	8.1	17.8	49.2	8.14	17.84	63.2	8.14	17.84	63.2	8.18	17.88	77.7	8.22	17.92	93.1
	451	256	2	451	256	3	226	256	3	451	256	4	226	256	4	151	256	4
	75.28	90.3	35.1	93.32	108.3	59.7	57.32	76.84	59.7	111.4	126.4	85.8	66.36	85.88	85.8	51.36	72.38	85.8
				1083	512	2	3564	1024	2	1083	512	3	1083	512	3	3564	1024	3
				184.1	217.5	47.5	558.7	609.3	59.6	227.4	260.8	78.9	227.4	260.8	78.9	701.3	751.8	97.3
										8037	2048	2	8037	2048	2	17807	4096	2
										1270	1374	71.7	1270	1374	71.7	2830	3045	81.1
200	1	256	3	1	256	4	1	256	5	1	256	5	1	256	6	1	256	7
	10.32	21.53	34.6	10.34	21.55	49.2	10.36	21.57	63.2	10.36	21.57	63.2	10.38	21.59	77.7	10.4	21.61	93.1
	1061	512	2	1061	512	3	531	512	3	1061	512	4	531	512	4	354	512	4
	89.83	106.3	35.1	111	127.5	59.7	68.65	90.38	59.7	132.3	148.7	85.8	79.27	101	85.8	61.57	85.06	85.8
				2500	1024	2	8165	2048	2	2500	1024	3	2500	1024	3	8165	2048	3
				214.8	250.9	47.5	643.5	696.1	59.6	264.8	300.9	78.9	264.8	300.9	78.9	806.8	859.4	97.3
										18227	4096	2	18227	4096	2	40091	8192	2
										1446	1554	71.7	1446	1554	71.7	3197	3419	81.1
300	1	512	3	1	512	3	1	512	4	1	512	4	1	512	5	1	512	5
	16.62	33.45	46	16.62	33.45	46	16.63	33.47	66	16.63	33.47	66	16.65	33.48	84.6	16.65	33.48	84.6
	2364	1024	2	7816	2048	2	2364	1024	3	2364	1024	3	2364	1024	4	2364	1024	4
	135.2	158.6	40	410	444.9	52.2	166.7	190.1	67.5	166.7	190.1	67.5	198.2	221.6	96.8	198.2	221.6	96.8
							5452	2048	2	17671	4096	2	5452	2048	3	5452	2048	3
							315.4	366.1	52.2	932.8	1005	64.3	388.1	438.8	86.1	388.1	438.8	86.1
													39114	8192	2			
													2076	2224	76.6			
500	1	512	5	1	512	7	1	512	8	1	1024	4	1	1024	4	1	1024	5
	10.39	20.49	32.3	10.4	20.5	44.8	10.41	20.51	51	25.62	49.9	74.7	25.62	49.9	74.7	25.63	49.9	95.3
	832	1024	2	624	1024	3	500	1024	3	5822	2048	3	5822	2048	3	9437	4096	3
	45.55	66.5	30.1	45.55	67.33	52	41.58	63.86	52	248.3	279.6	71.7	248.3	279.6	71.7	446.8	528.4	90.2
				5822	2048	2	18875	4096	2	41708	8192	2						
				201.7	233.1	42.7	597.8	641.6	54.9	1329	1419	67.1						
800	1	1024	3	1	1024	4	1	1024	5	1	1024	5	1	1024	6	1	1024	7
	15.19	29.17	34.6	15.19	29.17	49.2	15.2	29.18	63.2	15.2	29.18	63.2	15.2	29.18	77.7	15.21	29.19	93.1
	5472	2048	2	5472	2048	3	2734	2048	3	5472	2048	4	2734	2048	4	1823	2048	4
	118.4	137.3	35.1	145.8	164.6	59.7	91	116.7	59.7	173.1	192	85.8	104.7	130.4	85.8	81.89	109.9	85.8
				12509	4096	2	40291	8192	2	12509	4096	3	12509	4096	3	40291	8192	3
				273.3	314.2	47.5	800.7	856.5	59.6	335.8	376.7	78.9	335.8	376.7	78.9	1002	1058	97.3
1000	1	1024	5	1	1024	7	1	1024	8	1	2048	4	1	2048	4	1	2048	5
	12.36	23.54	32.3	12.36	23.55	44.8	12.37	23.55	51	29.67	56.11	74.7	29.67	56.11	74.7	29.68	56.11	95.3
	1868	2048	2	1401	2048	3	1121	2048	3	12881	4096	3	12881	4096	3	20748	8192	3
	52.07	74.77	30.1	52.07	75.7	52	47.59	71.78	52	276.4	309.6	71.7	276.4	309.6	71.7	493.6	580.2	90.2
				12881	4096	2	41503	8192	2									
				224.9	258.1	42.7	659.7	704.8	54.9									
2000	1	2048	5	1	2048	7	1	2048	8	1	4096	4	1	4096	4	1	4096	5
	14.36	26.61	32.3	14.37	26.61	44.8	14.37	26.61	51	33.75	62.29	74.7	33.75	62.29	74.7	33.75	62.3	95.3
	4138	4096	2	3105	4096	3	2484	4096	3	28211	8192	3	28211	8192	3	45182	16384	3
	58.56	82.97	30.1	58.58	84.02	52	53.61	79.67	52	304.2	339.2	71.7	304.2	339.2	71.7	539.8	631.1	90.2
				28211	8192	2												
				247.8	282.8	42.7												
5000	1	8192	3	1	8192	3	1	8192	4	1	8192	4	1	8192	5	1	8192	5
	28.87	51.79	44.6	28.87	51.79	44.6	28.87	51.79	63.9	28.87	51.79	63.9	28.87	51.79	82	28.87	51.79	82
10000	1	16384	3	1	16384	3	1	16384	4	1	16384	4	1	16384	5	1	16384	5
	32.14	56.71	44.6	32.14	56.71	44.6	32.14	56.71	63.9	32.14	56.71	63.9	32.14	56.71	82	32.14	56.71	82



(a) IFFT method



(b) finite difference method

Fig. 10. Complexity estimation via operation count ($c = 2m + a$) versus execution time per sample point. For both methods, the execution time lies mostly between $0.4c$ and $0.8c$ nanoseconds, indicated by straight lines.

Fig. 10 shows the results. For both methods, the execution time per sample lies mostly between $0.4c$ and $0.8c$ nanoseconds. Only for small O and small frame sizes, up to $1.2c$ is reached, but these are cases where the IFFT method should not be used anyway. The finite difference method converges quickly to $0.5c$, so its optimization gains an additional speedup between 0.8 and 1.6 over the theoretical performance comparison. As a result, the conclusion of Section VI, i.e. that the IFFT method is faster for $O > 20$ and $T > 50$, remains basically valid, albeit for slightly higher minimal O. A direct comparison ist difficult, due to the vast number of parameter combinations. However, the speedup can easily be approximated by $\frac{4O}{p(2m+a)}$ or simply $\frac{O}{pm}$ in Table I, where $0.8 < p < 1.6$.

Note that both methods can be accelerated by SIMD-extensions such as SSE. The FFTW-library is about 3 times faster by using SSE than the FFT used in the above measurements. The SIMD-potential of the finite difference scheme may still need exploitation [23].

VIII. Exact Border Match

The biggest approximation errors are located at the borders of the frame, introducing discontinuities at regular intervals. The following approach restricts the Fourier coefficients in a way so that the values at the border of the frames are exact.

The condition of exact border values is properly expressed

$$y(T_0) = x(T_0)h(T_0), \qquad y(T_1) = x(T_1)h(T_1)\,. \tag{20}$$

These two equations serve as constraint in the optimization of (6). By applying Lagrange-multipliers λ_0, λ_1 to these equations, the solution in (9) becomes

$$\sum_{k=M_0}^{M_1} Y(k)\tilde{G}(k-n) + \lambda_0 e^{-in\frac{2\pi}{N}T_0} + \lambda_1 e^{-in\frac{2\pi}{N}T_1} = \tilde{X}(n)\,. \tag{21}$$

Together with (20), where $y(T_j)$ is substituted as in (3), we get a system of equations of size $(M+2)\times(M+2)$, which can easily be solved as in the non-constrained case.

The approximation accuracy, however, is not improved by this approach, but decreases by several dB. On the other hand, errors in the high frequency range are transferred to lower frequencies, which may be preferable for some applications.

IX. Conclusion

The method of sinusoid synthesis via inverse FFT is usually implemented by an analysis part, i.e. choosing a window function, forward Fourier transform of windowed sinusoids, and dropping insignificant Fourier coefficients, followed by the synthesis part, i.e. inverse Fourier transform of the possibly relocated coefficients, and application of the inverted window function. The signal is then truncated at the borders to a certain frame size. Thus, computational complexity per oscillator is reduced and shifted to an FFT-part that is common to all synthesized oscillators.

This method can be significantly improved by two approaches. First, for a given number of Fourier coefficients of an oscillator, the values of the coefficients can be chosen optimally, so that the average approximation error over the whole range of frequencies is minimized. The optimization is achieved by the solution of a small system of linear equations. The gain in overall approximation SNR is 5 to 20 dB.

Second, an optimal window function can be found, which minimizes the approximation error for optimal coefficients computed as above. This is done with an iterative approach that converges to the optimal solution. The gain in overall SNR is another 5 to 20 dB, so the total improvement is up to 40 dB in practically relevant parameter ranges.

The applied methods can be modified to allow for amplitude changes and exact border matches. Also, a table has been generated which can be used to look up the optimal parameter choice, i.e. the optimal number of coefficients and FFT-size, for given frame size, minimal SNR and number of synthesized oscillators. This parameter choice minimizes the computational complexity of the sinusoid synthesis. It turns out that the truncation of the FFT-generated signal to frame size should be quite drastic if the number of oscillators is large, especially if a truncated FFT implementation can be used.

Appendix

A. Optimal Coefficients

To get (7), we have to minimize the approximation error (6) by varying the Fourier coefficients $Y(k)$. Because the error function contains non-holomorphic elements (norm or conjugation), we cannot build the derivative in a strict sense. However, the following approach amounts to practically the same. Suppose that $u(z)$ is a complex function of $z = x + iy$, i.e. $u(z) = v(x, y) + iw(x, y)$, where v, w, x, and y are real. If we want to get both derivatives $\frac{d}{dx}|u|^2 = \frac{d}{dx}(v^2 + w^2)$ and $\frac{d}{dy}|u|^2 = \frac{d}{dy}(v^2 + w^2)$, we can use the theorem

$$\frac{d|u|^2}{dz} := \frac{d|u|^2}{dx} + i\frac{d|u|^2}{dy} = 2u\overline{\frac{du}{dz}}\,. \tag{22}$$

Thus, we get both derivatives as real and imaginary part of a complex expression which constitutes a variation of the chain rule. Setting this expression to zero will set both derivatives to zero, as is intended. For our error function, this means

$$\begin{aligned}\frac{d}{dY(n)} \sum_{t=T_0}^{T_1} |g(t)y(t) - x(t)|^2 \\ = \sum_{t=T_0}^{T_1} 2(g(t)y(t) - x(t))g(t)\overline{\frac{dy(t)}{dY(n)}}\,. \end{aligned}\tag{23}$$

Now, we substitute $y(t) = \sum_{k=M_0}^{M_1} Y(k)e^{ik\frac{2\pi}{N}t}$ as in (3) and get

$$2\sum_{t=T_0}^{T_1}\left(g(t)\sum_{k=M_0}^{M_1} Y(k)e^{ik\frac{2\pi}{N}t} - x(t)\right)g(t)e^{-in\frac{2\pi}{N}t}\,. \tag{24}$$

Setting this expression to zero, finally results in (7).

B. Optimal Window Function

To minimize (10) to get (11), we choose a similar approach as in Appendix A. First, we build the derivative of (10):

$$\begin{aligned}\frac{d}{dg(t)}\int_{\alpha_0}^{\alpha_1}\sum_{s=T_0}^{T_1} |g(s)y(\alpha, s) - x(\alpha, s)|^2 d\alpha \\ = \int_{\alpha_0}^{\alpha_1} 2(g(t)y(\alpha, t) - x(\alpha, t))\overline{y(\alpha, t)}\,.\end{aligned}\tag{25}$$

Then we set this expression to zero and arrive at (11).

References

[1] T. F. Quatieri and R. J. McAulay, "Audio signal processing based on sinusoidal analysis/synthesis," in *Applications of digital signal processing to audio & acoustics*. Kluwer Academic Publishers, 1998, pp. 343–416.

[2] P. Kleczkowski, "Group additive synthesis," *Computer Music J.*, vol. 13, no. 1, pp. 12–22, 1989.

[3] J. S. Marques and L. B. Almeida, "Frequency-varying sinusoidal modeling of speech," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, pp. 763–765, May 1989.

[4] X. Serra and J. O. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music J.*, vol. 14, pp. 12–24, 1990.

[5] M. Lagrange and S. Marchand, "Real-time additive synthesis of sound by taking advantage of psychoacoustics," in *Proc. Digital Audio Effects (DAFx) Conf.*, 2001, pp. 5–9.

[6] N. Meine and H. Purnhagen, "Fast sinusoid synthesis for MPEG-4 HILN parametric audio decoding," in *Proc. Digital Audio Effects (DAFx) Conf.*, Sep. 2002, pp. 239–244.

[7] J. W. Gordon and J. O. Smith, "A sine generation algorithm for VLSI applications," in *Proc. Intern. Computer Music Conf. (ICMC)*. Computer Music Association, 1985, pp. 165–168.

[8] M. Robine, R. Strandh, and S. Marchand, "Fast additive sound synthesis using polynomials," in *Proc. Digital Audio Effects (DAFx) Conf.*, 2006, pp. 181–186.

[9] J. Bonada and X. Serra, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Process. Mag.*, vol. 24, no. 2, pp. 80–91, Mar. 2007.

[10] E. Lindemann, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Process. Mag.*, vol. 24, no. 2, pp. 67–79, Mar. 2007.

[11] X. Rodet and P. Depalle, "A new additive synthesis method using inverse Fourier transform and spectral envelope," in *Proc. Intern. Computer Music Conf. (ICMC)*, 1992, pp. 410–411.

[12] M. Goodwin and A. Kogon, "Overlap-add synthesis of nonstationary sinusoids," in *Proc. Intern. Computer Music Conf. (ICMC)*, 1995, pp. 355–356.

[13] E. B. George and M. J. T. Smith, "Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones," *J. Audio Engineering Society*, vol. 40, no. 6, pp. 497–516, 1992.

[14] J. O. Smith and X. Serra, "PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation," in *Proc. Intern. Computer Music Conf. (ICMC)*, 1987.

[15] J. Harris, "On the use of windows for harmonic analysis with the descrete Fourier transform," *Proc. IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[16] J. Laroche, "Synthesis of sinusoids via non-overlapping inverse Fourier transform," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 4, pp. 471–477, Jul. 2000.

[17] M. D. Sacchi, T. J. Ulrych, and C. J. Walker, "Interpolation and extrapolation using a high-resolution discrete Fourier transform," *IEEE Trans. Signal Process.*, vol. 46, no. 1, pp. 31–38, Jan. 1998.

[18] A. Papoulis, "A new algorithm in spectral analysis and band-limited extrapolation," *IEEE Trans. Circuits Syst.*, vol. 22, no. 9, pp. 735–742, 1975.

[19] R. Strandh and S. Marchand, "Real-time generation of sound from parameters of additive synthesis," in *Proc. Journées d'Informatique Musicale (JIM)*, 1999, pp. 83–88.

[20] J. O. Smith and P. R. Cook, "The second-order digital waveguide oscillator," in *Proc. Intern. Computer Music Conf. (ICMC)*, 1992, pp. 150–153.

[21] J. van der Hoeven, "The truncated Fourier transform and applications," in *ISSAC '04: Proc. Intern. Symp. Symbolic and Algebraic Computation*. ACM, Jul. 2004, pp. 290–296.

[22] R. Kutil, "Webpage with audio examples and further material," `http://www.cosy.sbg.ac.at/~rkutil/oscarr.html`.

[23] T. Hodes and A. Freed, "Second-order recursive oscillators for musical additive synthesis applications on SIMD and VLIW processors," in *Proc. Intern. Computer Music Conf. (ICMC)*, 1999, pp. 74–77.