Digital Sound Generation

Beat Frei
Institute for Computer Music and Sound Technology (ICST)
Zurich University of the Arts
Baslerstrasse 30, CH-8048 Zürich, Switzerland
beat.frei@zhdk.ch, http://www.icst.net

Preface

This online book presents advanced techniques for digital sound generation in electronic musical instruments with focus on the discussion and realization of popular building blocks using industrial grade algorithms.

It has been written to serve as a reference for synthesizer development as well as to support university level courses on audio signal processing and computer music.

The reader is assumed to be familiar with the basics of sound synthesis and signal theory. While a graphical time-frequency viewpoint based on convolution and sampling is emphasized to explain the foundations, analysis and implementation occasionally involve higher mathematics or technical computing tools.

Happy reading!

Table of Contents

1 Main Oscillators

1.1 Generic Oscillator and Definitions

Many digital oscillators share a generic topology: A phase accumulator generates a simple sawtooth with frequency and phase control, a frequency dependent memoryless function maps it to the desired shape, and an optional postfilter tailors the high frequency spectrum (Fig.1).

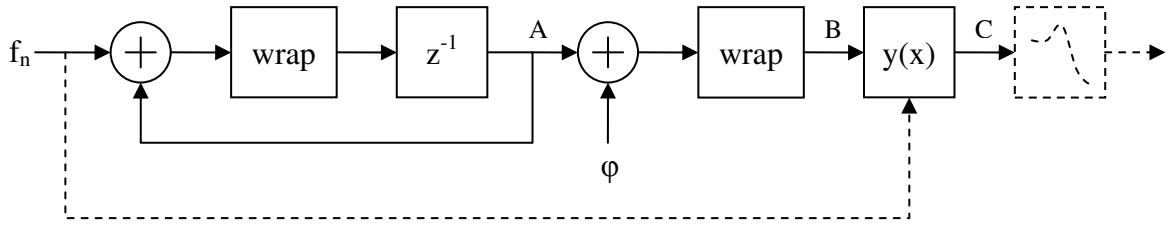


Fig. 1: Generic Digital Oscillator Topology

$\mathbf{f_o}$	=	**Fundamental Frequency**
$\mathbf{f_s}$	=	**Sample Rate** (Default: 48 kHz. Double rate systems: 96 kHz.)
T	=	**Sampling Interval = $1/f_s$**
$\mathbf{f_n}$	=	**$2Nf_o/f_s$** (Normalized Fundamental Frequency, ubiquitously used)
φ	=	**Phase shift in radians · N/π**
$\mathbf{z^{-1}}$	=	**Delay of one T.** Output changes in discrete time steps.
wrap(x)	=	**x-2N if x ≥ N, x+2N if x < -N, x otherwise, repeat until the result is within [-N, N)** (Default: N = 1) Important: Wrap(x) is inherent to the addition of integer data types in processors. No further operations are required if the two's complement wrap-around is exploited by setting N to $2^{(\text{word size of data type in bits - 1})}$.
y(x)	=	Full cycle of a mapper function. Examples: Sine, wavetables selected according to f, polynomial y(x) with coefficients c(f).

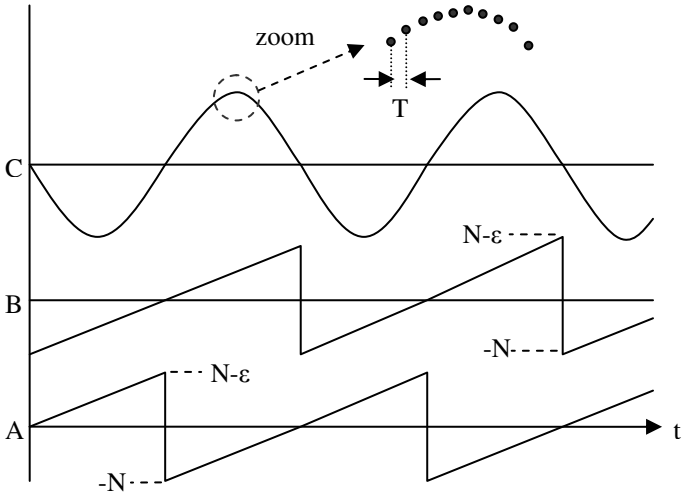


Fig. 2: Signals in the Generic Oscillator for y = sin(πx/N), φ = -N, no Postfilter

All signals are shown from a macroscopic perspective. Actually, they are number sequences changing in discrete time steps of the sampling interval T. In the next section we are going to discuss what difference that makes.

1.2 Analog versus Digital

Unlike analog circuitry, digital oscillators are discrete time systems and therefore have a spectrum periodic in the sample rate f_s. It is found by viewing the output as a sampled version of a continuous time signal, whose two-sided spectrum is superimposed at integer multiples of f_s by the sampling process. A component at f_o in the continuous signal leads to additional ones at $|Nf_s \pm f_o|$ with N integer in the oscillator output. If these artefacts fall into the audio band, aliasing occurs that is generally not removable and perceived as unwanted tones. Fig. 3 shows a continuous time sawtooth and its discrete time representation including aliasing, whose psychoacoustical relevance will be discussed on the basis of selected components a-d.

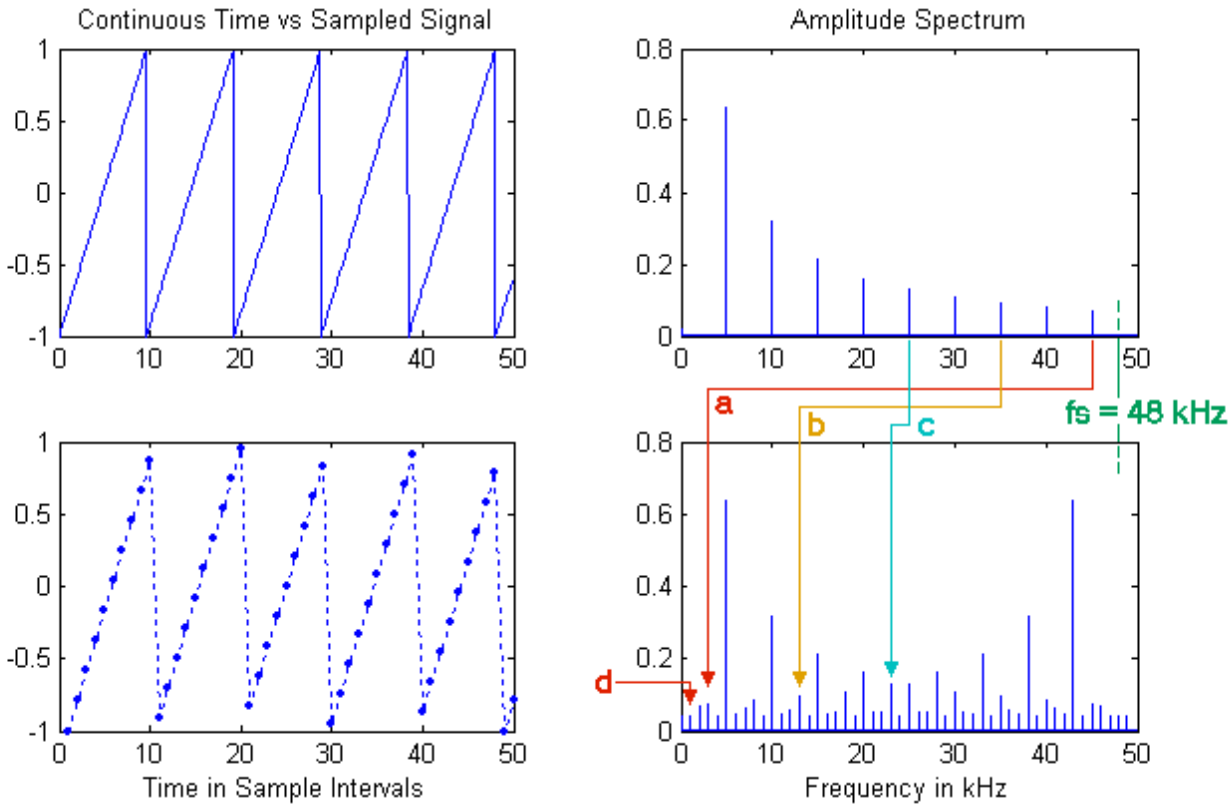


Fig. 3: Continuous Sawtooth with f_o = 5 kHz sampled at f_s = 48 kHz

In practice, some aliasing is tolerable due to negligible signal power and auditory masking: In the presence of a strong tone, higher pitched weak tones are inaudible. For lower pitched tones, this effect diminishes rapidly and becomes negligible at intervals larger than an octave (Fig. 4). A coarse analysis of the sampled sawtooth from an auditory perspective yields:

a) Aliased 9th harmonic of the original: $f_s - 9f_o$ = 3 kHz. Audible.
b) Aliased 7th harmonic of the original: $f_s - 7f_o$ = 13 kHz. Partially masked by the 2nd harmonic of the original. Human pitch perception is imprecise at the top of the audio band, so this component has a chance to pass as "general highs". Your ears decide.
c) Aliased 5th harmonic of the original: $f_s - 5f_o$ = 23 kHz. Inaudible.
d) Aliased 19th harmonic of the original: $2f_s - 19f_o$ = 1 kHz. Audible. An ad-hoc transposition of Fig. 4 suggests that this component should be at least 80 dB down relative to the original to become inaudible.

Conclusion: Spectral components of the continuous original that are close to integer multiples of f_s will lead to audible and objectionable low frequency aliasing when the signal is sampled. The effect is most distinct for aliased components falling below the fundamental frequency f_o.

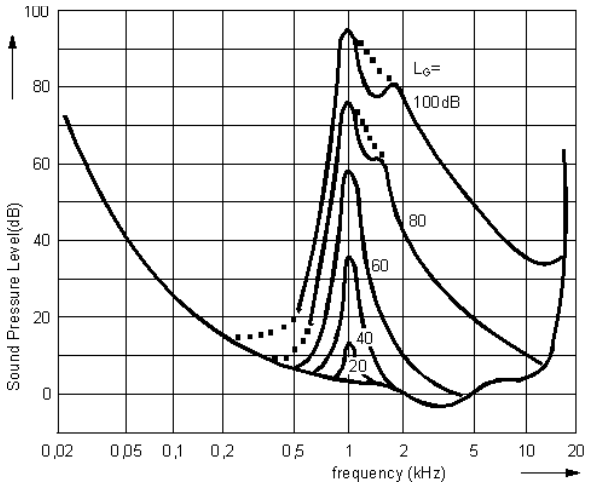


Fig. 4: Auditory Threshold in the Presence of a Narrow Band Signal at 1 kHz
Ref.: E. Zwicker, R. Feldtkeller: Das Ohr als Nachrichtenempfänger. (1967)

Practical experience shows that a synthesizer should be able to generate tuned sounds with a fundamental up to 4-5 kHz at excellent quality, whereas higher frequency signals are mainly used as less critical modulators or sources of harmonics. Summarizing the foregoing we can state an empirical design rule for a continuous time prototype signal (see also Fig. 5):

- Desired spectrum up to 20 kHz.
- Roll-off as steep as possible within given constraints.
- Below -80 dB at f_s – 2 kHz
- Below -85 to -90 dB at f_s – 1 kHz
- Roll-off from f_s – 1 kHz to infinity by at least -20 dB/decade (≈ 6 dB/octave)

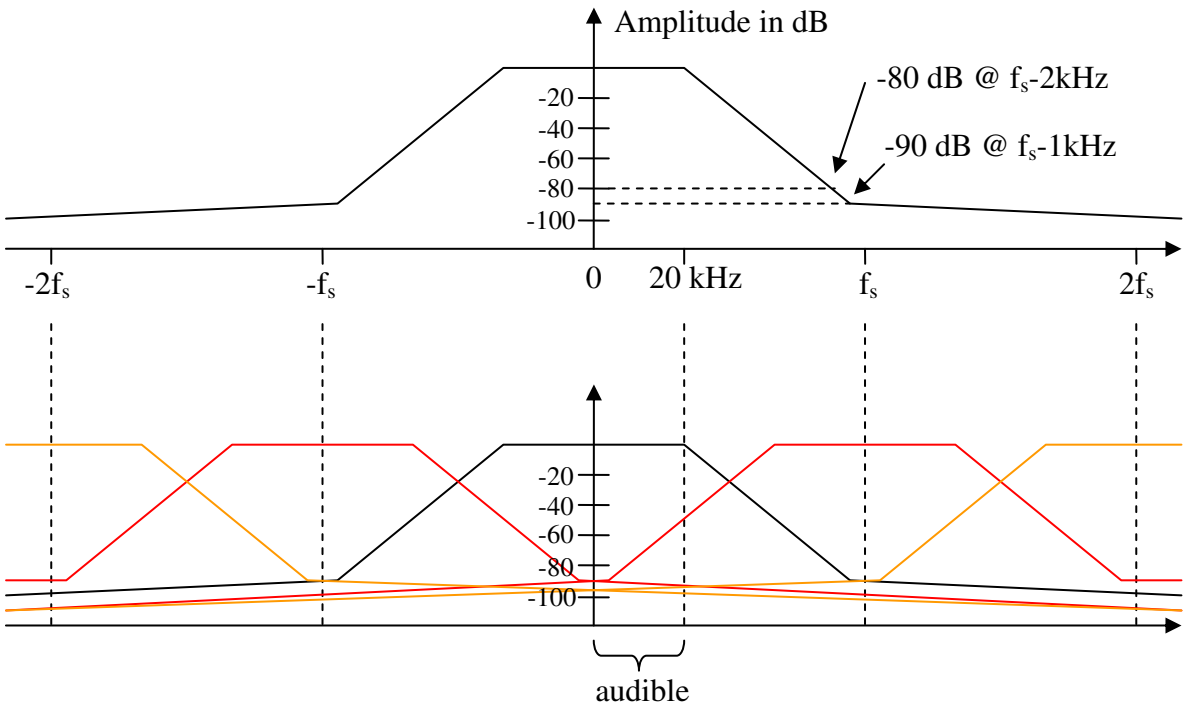


Fig. 5: Spectrum of a continuous time prototype signal and its discrete time representation including 1st and 2nd order aliased components (red, orange)

1.3 Computation-Friendly Bandlimited Impulse

Now that the requirements for a continuous time prototype signal are known, we choose the most elemental function that meets them and derive more complicated signals from it.

A good starting point is the Dirac delta impulse with its flat spectrum up to infinity. When we send it through a filter with the frequency response of Fig. 5, the output spectrum is an exact copy of the response while the signal itself is the sought-after bandlimited impulse b(t). Since the computational effort for many algorithms based on b(t) is proportional to its length in sampling intervals T, it should be as short as possible.

Theoretically, we could use a brickwall filter with infinite attenuation outside the audio band, but the resulting impulse would be the slowly decaying Sinc function. A popular solution is to limit the impulse in time by weighting it with a finite-length window function [11]:

1. Take the impulse response of a brickwall filter with a cut-off frequency f_c = 20 kHz and the sample rate f_s: $g(t) = Sinc(\pi f_c / f_s) = \frac{\sin(\pi f_c / f_s)}{(\pi f_c / f_s)}$
2. Choose a parametric window w(t,β) with low side lobes.
3. Calculate b(t) = w(t, β)g(t) and its spectrum.
4. Repeat steps 1 to 3 tweaking β and f_c until you get the desired stopband attenuation.
5. Adjust f_c slightly to maximize the attenuation around f_s.
6. Consider apodization (using a second window that widens the top of the main window) to tailor the transition band.
7. (Optional: Resort to the Parks-McClellan algorithm for utmost performance.)

Examples for lengths 4T and 10T with T = 1/f_s and f_s = 48 kHz using a Kaiser window are given in Fig. 6-8. Refer to appendix B1 for MATLAB code.

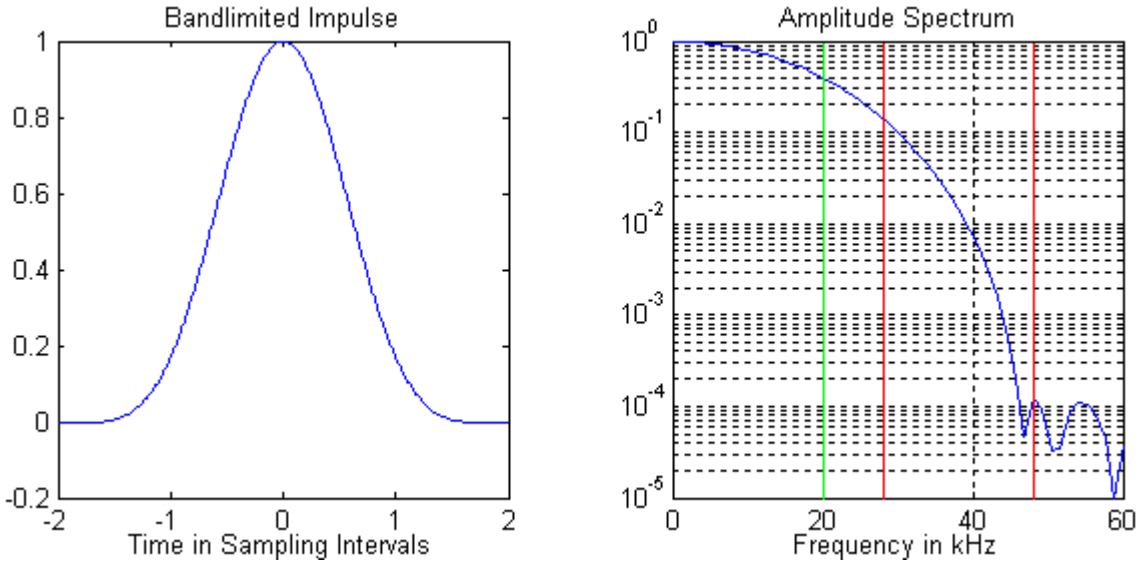


Fig. 6: Bandlimited Impulse (Length = 4T, f_s = 48 kHz, f_c = 15 kHz, normalized)
Main Window: Kaiser (β = 8.3), Apodizing Window: 1- 0.5·Kaiser (β = 0.5)

The signal shown in Fig. 6 is suitable for virtual analog oscillators where compactness is crucial. Since not the impulse itself but its time integral with less high frequency content is actually used, the requirements for stopband attenuation are reduced.

We have to equalize the rounded passband edges to avoid a dull sound. A postfilter or preemphasis works best if the drop does not exceed 10 dB at 20 kHz, however, this is not overly critical. We might conservatively lower f_c in the 4-sample impulse to further reduce susceptibility to aliasing and accept stronger filter action.

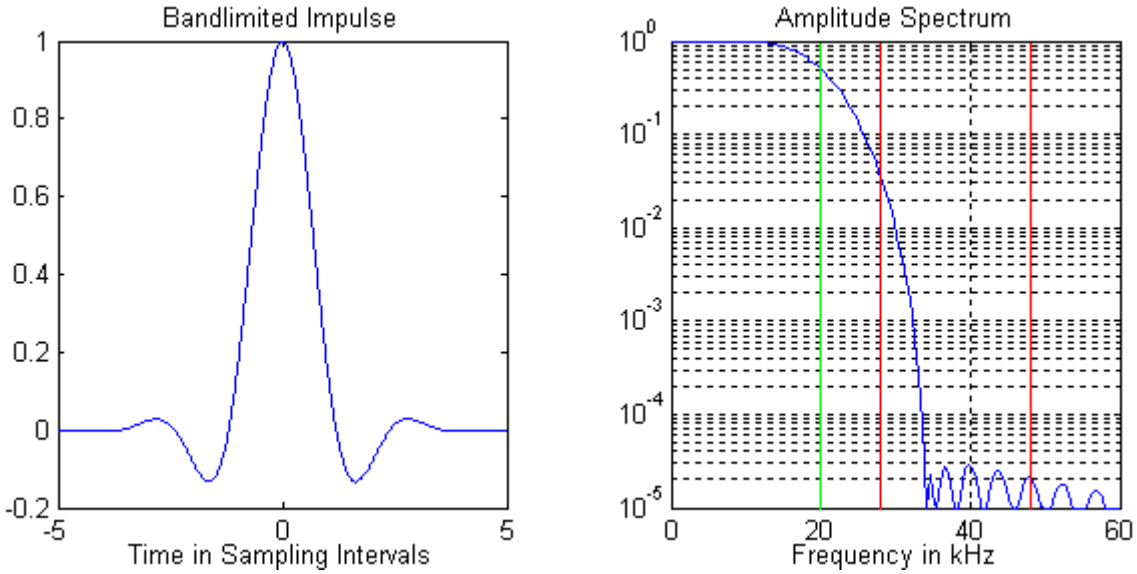


Fig. 7: Bandlimited Impulse (Length = 10T, f_s = 48 kHz, f_c = 20 kHz, normalized)
Main Window: Kaiser (β = 9), no Apodization

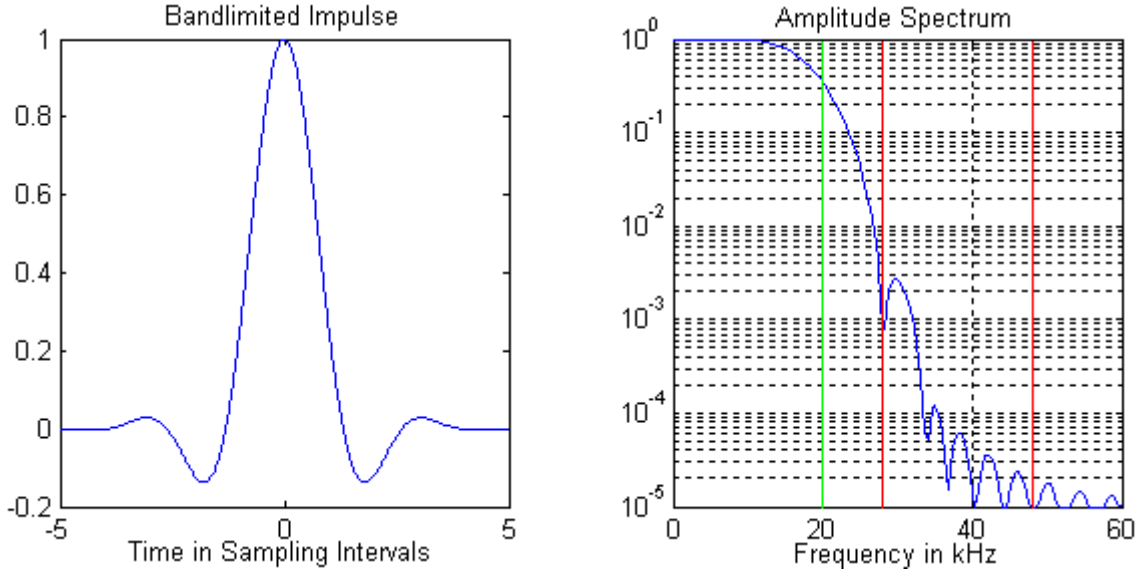


Fig. 8: Bandlimited Impulse (Length = 10T, f_s = 48 kHz, f_c = 18.3 kHz, normalized)
Main Window: Kaiser (β = 9), Apodizing Window: 1- 0.9·Kaiser (β = 0.7)

A comparison of Fig.7 and 8 shows the effect of apodization. Higher attenuation at the beginning of the stop band at the expense of a slower roll-off results in a spectral profile which is especially suitable for sample-based oscillators.

So far, we looked at continuous time impulses. As they are hard to calculate and polynomial approximations lead to a large number of terms, it seems favourable to tabulate them. A single table look-up is very attractive compared to interpolation-based techniques with their typically

3 to 5 times higher operations count and 2 to 3 times longer processing time on contemporary computing platforms. However, this approach is often deemed impractical, which is true for systems intended to process arbitrary audio signals. But, how does it perform under the specific requirements for oscillators in musical instruments?

First, we look at the spectral difference between a continuous signal and its tabulated representation assuming that a table read is performed by truncating or rounding a continuous time pointer to form an integer index. N denotes the number of table entries per sampling interval T.

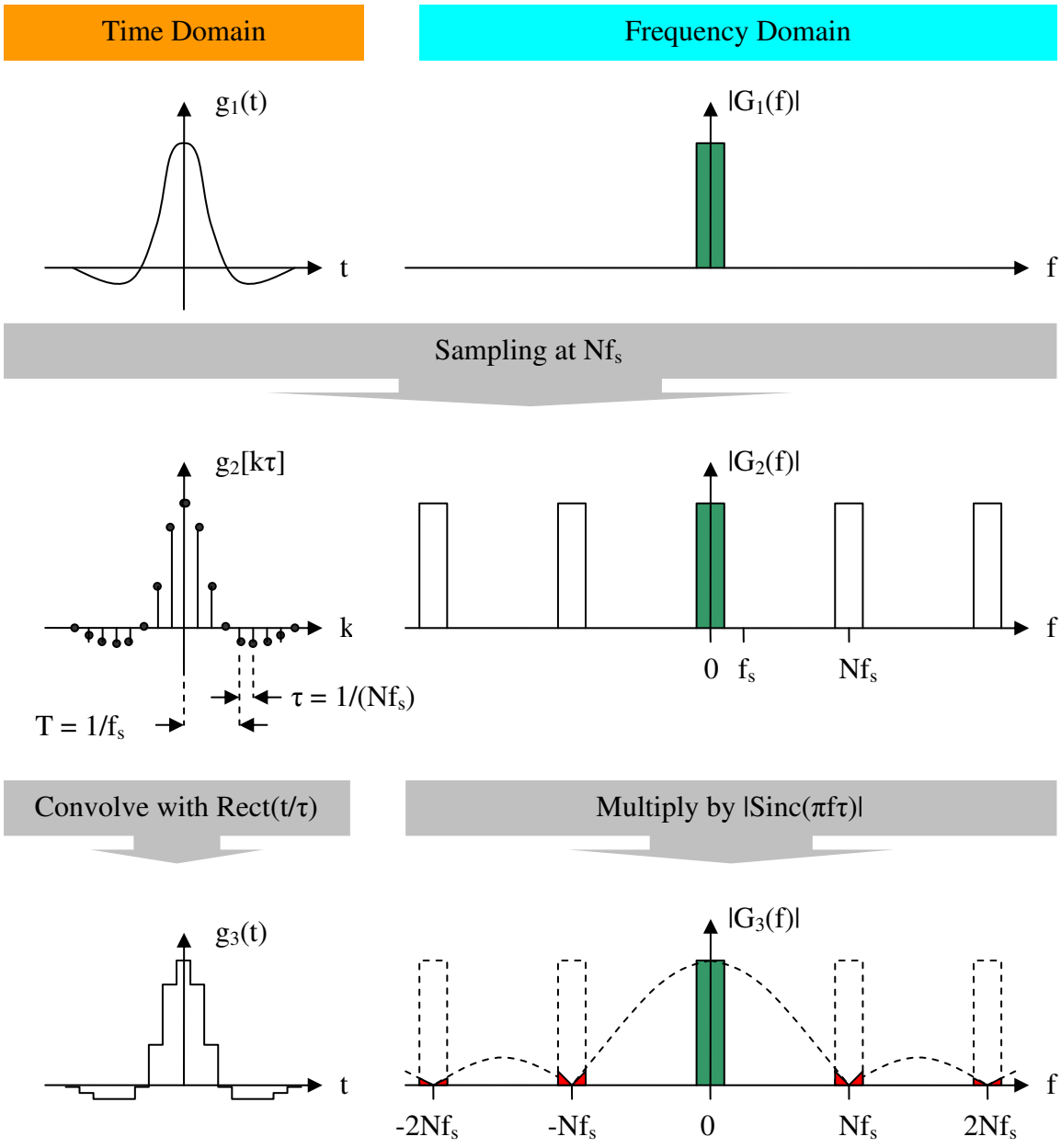


Fig. 9: Spectral consequences of converting a continuous signal to a table intended for nearest neighbor look-up, N = 4 table entries per sampling interval T

Reading the table at a certain rate f_r in a digital oscillator corresponds to sampling the signal $g_3(t)$ at f_r, hence spectral components around integer multiples of f_r are fold down into the audio band. In the next step, we calculate the relative amount of aliasing as a root mean square ratio γ for a component of the frequency f_o in the continuous non-tabulated original under the condition $f_r \approx f_s$.

A direct evaluation using $\pi f_o \tau << 1$ as required in any practical application yields for each of the two components aliased down from kNf_s:

$$\gamma_s = \left| \frac{Sinc(\pi\tau[kNf_s \pm f_o])}{Sinc(\pi f_o \tau)} \right| \approx \left| \frac{\sin(\pi\tau[kNf_s \pm f_o])}{\pi\tau[kNf_s \pm f_o]} \right| = \frac{\sin(\pi f_o \tau)}{\pi\tau[kNf_s \pm f_o]} \approx \frac{f_o}{kNf_s}$$

Aliased components can be assumed to have non-coincidental frequencies. Therefore, we sum up their power to get an upper bound for the total amount:

$$\gamma = \frac{|G_3(aliased)|_{RMS}}{|G_3(f_o)|} \leq \sqrt{\sum_{k=1}^{\infty} 2\left[\frac{f_o}{kNf_s}\right]^2} = \frac{\pi}{\sqrt{3}} \cdot \frac{f_o}{Nf_s} \approx \frac{1.8 f_o}{Nf_s} \qquad \text{(Eq. 1)}$$

Eq. 1 paves the way to determine the minimum required size of a look-up-table from the spectrum of the non-tabulated original signal. Although we stated $f_r \approx f_s$, it's not hard to see that it also provides a safe estimate for $f_r > f_s$. In case of $f_r < f_s$, the components around kNf_s may be aliased down multiple times resulting in a worst case increase of 3 dB for doubling the ratio f_s/f_r. In practice, this is rarely important, because aliased components arising from undersampling of the base band will almost always dominate.

Summary: Aliasing from reading tabulated signals without interpolation affects the entire audio band. It should be kept below -85 to -90 dB in musical instruments according to Fig. 4. The amount generated by a spectral component of the original signal is proportional to its frequency and amplitude.

Example 1: Professional sample rate converter. 100 dB SNR for any signal up to 20 kHz, M = 100 filter taps. Table size = $MN = 1.8Mf_o/(\gamma f_s) = 7500000$, impractical.

Example 2: Sample-based oscillator. 97 dB SNR for an original component at 1 kHz and 85 dB for one at 4 kHz. Components at higher frequencies have lower levels in tuned sounds while noisy sounds are less critical with respect to perceived aliasing. When the sampled signal gets weaker, the aliased part proportionally decreases (in contrary to the constant noise floor of a converter or linear audio circuitry). Impulse length = 10T. Table size ≈ 26500, can be halved exploiting symmetry, fits into a CPU cache or the internal memory of a low-cost DSP.

Now that we know how to create bandlimited impulses and store them adequately, we are finally ready to design some oscillators!

1.4 Sample-Based Oscillator

This oscillator class works by playing a frequency-transposed version of a sampled sound. It's still the most common technique to emulate genuine musical instruments in a synthesizer. A big plus is the precise reproduction of any audio snapshot; major drawbacks are a repetitive character and the inability to inherently produce natural transitions.

The principles of work are straightforward: Consider a signal of frequency f_o and duration t_o in the original data that has been sampled at f_s corresponding to an interval $T = 1/f_s$. In the oscillator, the data is reconstructed, resampled at an interval $T_{new} = T/2$, and played back at f_s. As a result, the duration increases to $2t_o$ while the number of cycles remains the same. Thus, the frequency of this time-stretched signal has changed to $f_o/2$. Fig. 10 depicts the procedure for an arbitrary transposition factor $\alpha = f_{new}/f_o$.

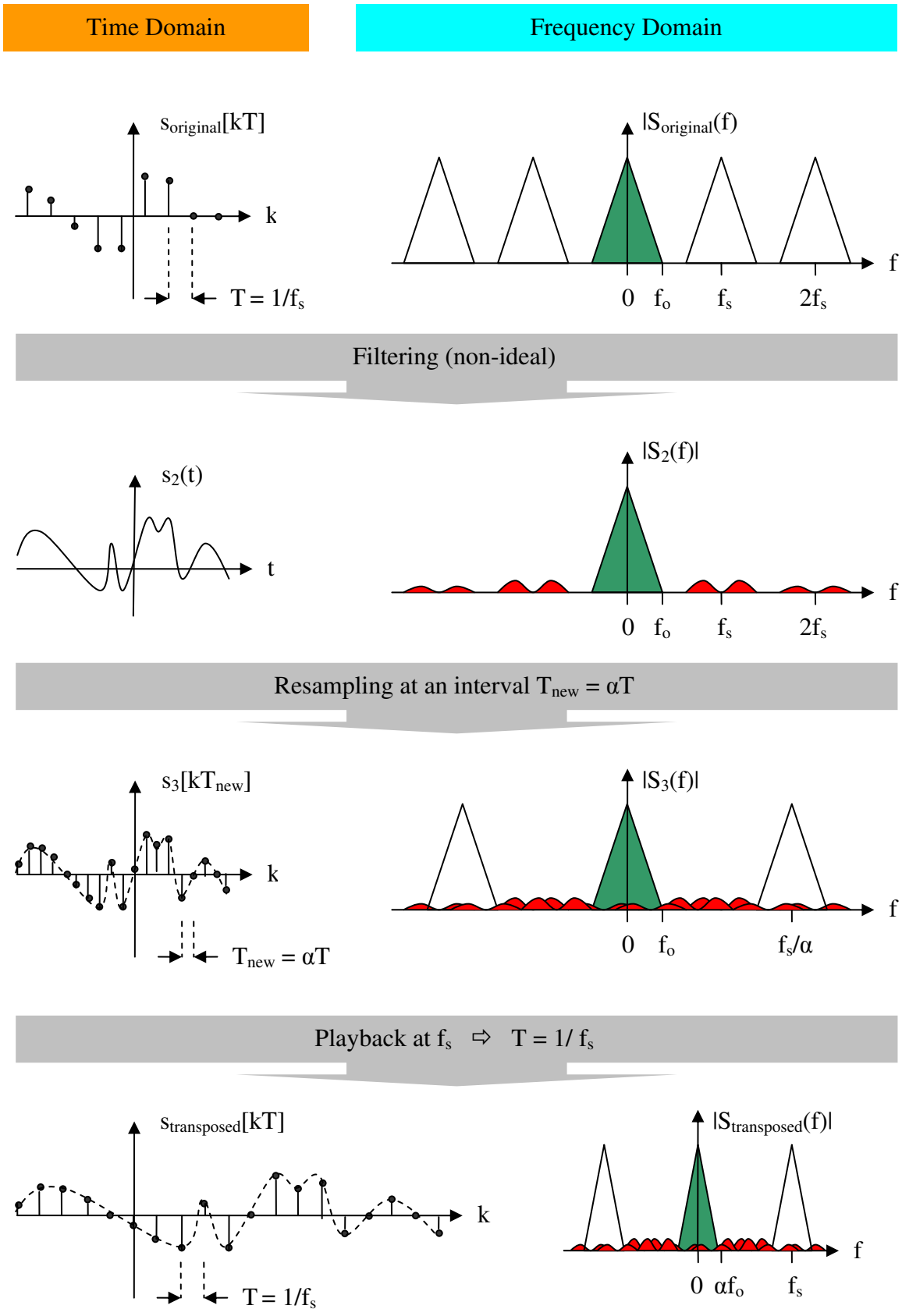


Fig. 10: Down Transposition in a Sample-Based Oscillator including Aliasing from Non-ideal Filtering

Fig. 10 allows us to analyze the limitations and requirements of a sample-based oscillator.

First, we have a look at down transposition ($\alpha < 1$):
If the filtered signal does not contain any significant components above $f_s - f_o$, there will be no audible aliasing in the transposed signal. The worst case occurs for $\alpha \approx 1$ where the requirements become the same as for the bandlimited impulse. Thus, the spectral response of any of the impulses designed in section 1.3 will fit our needs. This is important as filtering will be achieved in the oscillator by convolving such an impulse with the original signal, which corresponds to multiplying the spectrum of the original signal with the impulse spectrum. Another point is that the original spectrum usually gets limited to 20 kHz in the recording process. Since we expect a minimum bandwith of 15 kHz for high quality audio, α must be at least 0.75.

For up transposition ($\alpha > 1$), the situation is more complicated (Fig. 11):
At $f_s/\alpha < 2f_o$, aliasing is inevitable with a computational-friendly constant length impulse, but transposition may still move it out of the audio band. The limiting condition becomes $f_s - \alpha f_o > 20$ kHz, restricting α to below 1.4 for $f_o < 20$ kHz and $f_s = 48$ kHz.

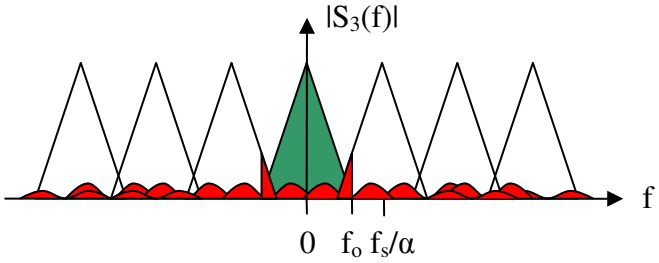


Fig. 11: Aliasing in Up Transposition

In practice, some aliasing is tolerated as it enters the audio band at the top and subsides with increasing α. In addition, tuned sounds tend to have low energy at high frequencies. However, aliasing that falls below the fundamental (max. 4 kHz) quickly becomes objectionable, which is the case for $f_s/\alpha - f_o < 4$ kHz. For $f_o = 20$ kHz and $f_s = 48$ kHz, a range of $\alpha = 1.4$ to 2 results. (Higher factors are accessible easiest by holding near-ideally lowpass filtered and up transposed versions of the original audio data in memory [18].) At last, the filter in the first step must have enough attenuation not to compromise quality taking the spectral roll-off of tuned sounds into account. Example: For $f_s = 48$ kHz and $\alpha = 1.4$, an original component at 13.7 kHz is filtered at 34.3 kHz and aliased to 0 kHz in the transposed signal. This component can safely be estimated at least 20 dB down relative to the fundamental in the original signal, hence the filter specifications are relaxed towards higher frequencies. The spectral profile in Fig. 8 has proven to perform very well in musical applications.

The following part deals with the implementation of the oscillator. Interestingly, the steps in Fig. 10 can be executed all at once using a bandlimited impulse! A non-interpolated table-based impulse requires about 20 kWords of memory and provides optimum speed. If the hardware runs out of fast memory, consider a linearly interpolated table (≈ 500 elements) at the expense of roughly doubling the operations count [18]. In case both sample look-ups and memory are costly, polynomial interpolation [17] and Farrow filters are viable alternatives.

The oscillator works as shown in Fig. 12:
Filtering is accomplished by convolving a bandlimited impulse with the sampled original data. The resulting continuous function is then evaluated at the resampling points. Luckily, there's no need to leave the discrete time domain as the above procedure is equivalent to sliding the impulse over the original signal and calculating a sum of products at these points.

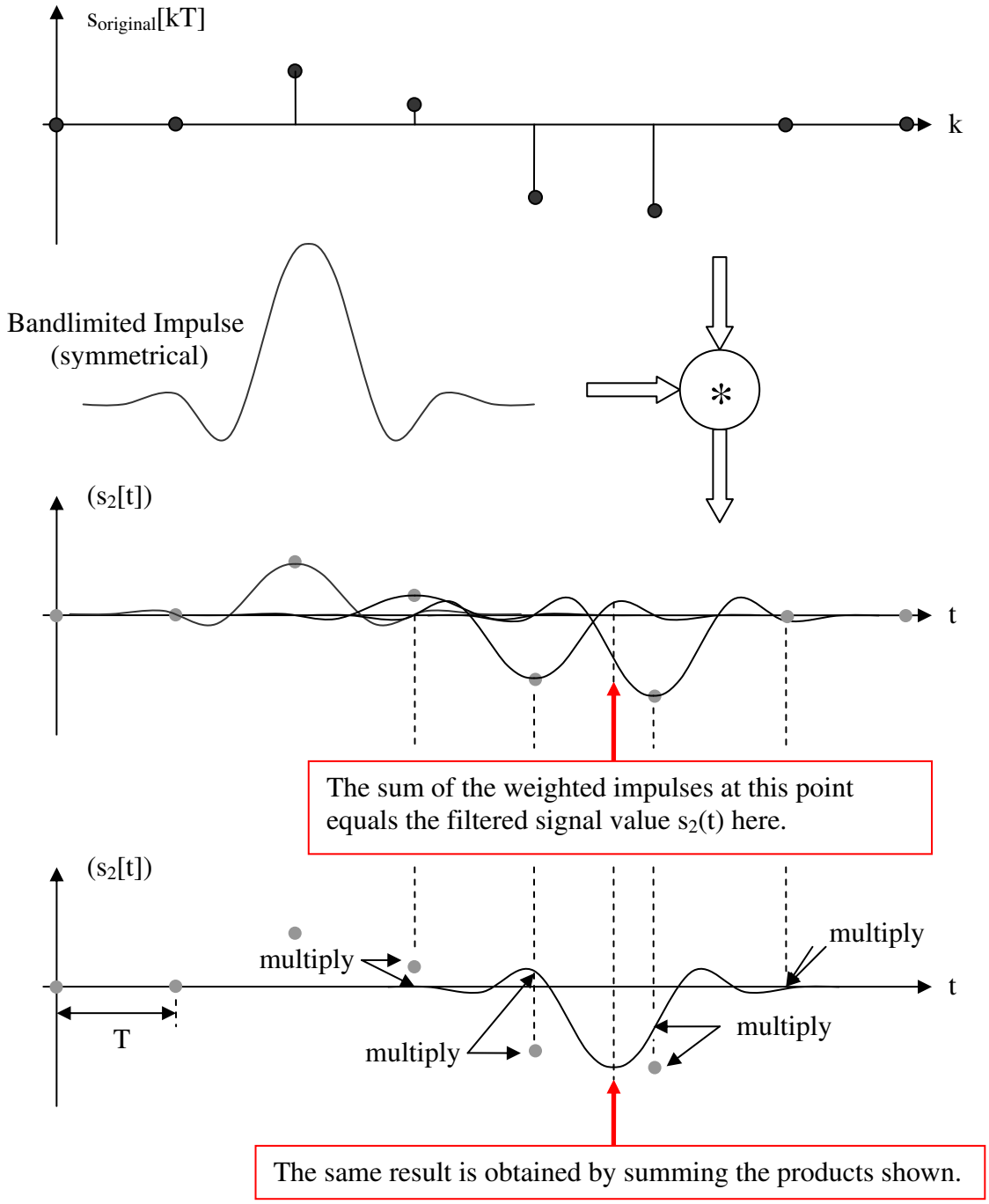


Fig. 12: Sliding Window Process of a Sample-Based Oscillator

To generate the output of the oscillator, we just advance the impulse in intervals αT and evaluate the sum of products. In the actual algorithm, the bandlimited impulse of Fig. 8 is read from a table. Hence, we need to calculate its minimum size using Eq. 1. If we aim to keep table-induced aliasing below -85 dB for a component at 4 kHz in the original data, we obtain:

$N = 10 \cdot 1.8 \cdot 4000/(48000 \cdot 5.6 \cdot 10^{-5}) = 26786$

For the oscillator algorithm to be efficient, the table must have 2^M entries per sampling interval. Therefore, we relax the requirements somewhat and choose $N = 10 \cdot 2048 = 20480$ which results in a noise level of -83 dB @ 4 kHz and -95 dB @ 1 kHz relative to the desired signal. Because the impulse is symmetrical, only 10240 entries have to be stored.

In the aforementioned calculation, Eq. 1 has been applied to the spectral product of the original sampled data and the tabulated bandlimited impulse. Fig. 13 justifies this approach provided that the number of table entries N per sampling interval is integer.

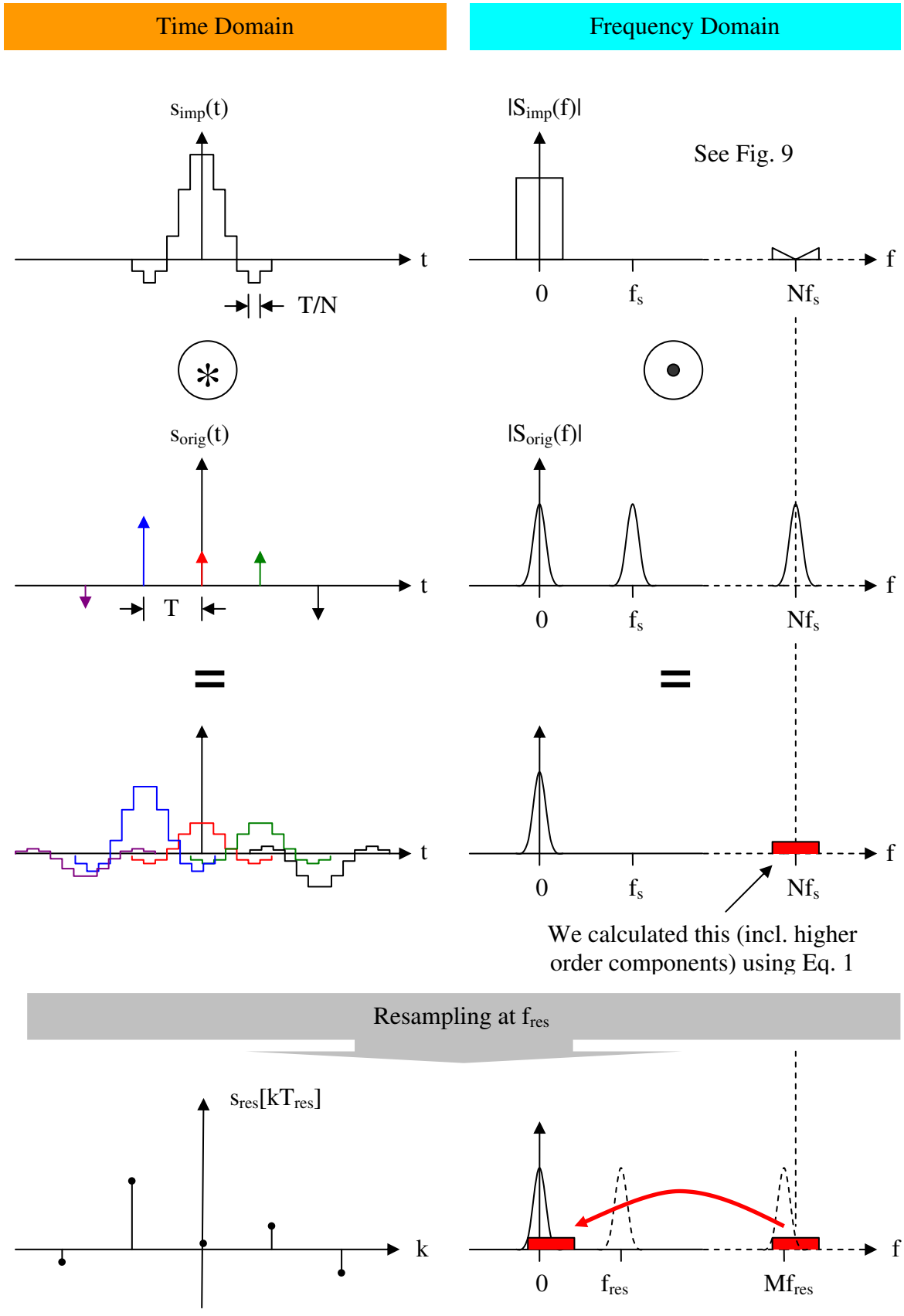


Fig. 13: Table-Induced Aliasing in the Sample Transposition Process

One aspect not yet addressed is the high frequency attenuation of the bandlimited impulse whose spectrum weights the original sampled data in the filtering process. Since that happens before resampling, the transposition ratio has no influence. Applying a linear-phase FIR prefilter to the sampled data when it is loaded into memory works well and avoids additional runtime processing. An example design for sampling at 48 kHz combines the normalized impulse of Fig. 8 with the prefilter below to get a mere ±0.05 dB gain deviation and 3.09 dB overall gain. Refer to appendix B2 for Matlab code based on Parks-McClellan optimization.

$$y[k] = \sum_{n=0}^{16} c_n x[k+n]$$

$c_0 = c_{16} = 0.0028$
$c_1 = c_{15} = -0.0119$
$c_2 = c_{14} = 0.0322$
$c_3 = c_{13} = -0.0709$
$c_4 = c_{12} = 0.1375$
$c_5 = c_{11} = -0.2544$
$c_6 = c_{10} = 0.4384$
$c_7 = c_9 = -0.6334$
$c_8 = 1.7224$

(Prefilter)

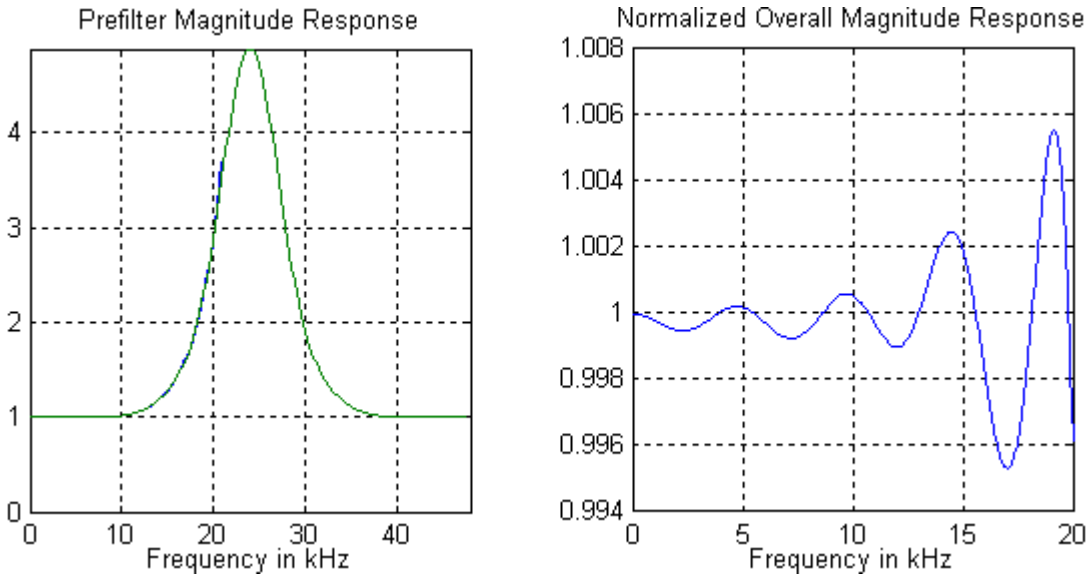


Fig. 14: Magnitude Response of the Combined Design "Fig. 8 and Prefilter"

We finish the section presenting the sliding window algorithm as a C snippet, which also justifies the decision to restrict the number of table entries per interval to a power of two. Useful modifications would be: 64-bit integers to support long samples, table interleaving in order to read from consecutive memory locations to improve caching on PC architectures.

```
float blimp[10*2048];                    // tabulated bandlimited impulse acc. to Fig. 8
float samples[length];                   // prefiltered original audio sample
unsigned int ptr, ratio;                 // sample pointer * 2048, transpose ratio (2048 ⇔ 1:1)
unsigned int index, offset ;

//*** calculate interpolated output value and advance in time***
index = ptr >> 11; offset  = 2047 + (index << 11) - ptr;     // much faster than using ptr/2048 and ptr%2048
out = 0.0f;
for (i = 0; i++; i < 10) {
        out += blimp[offset]*samples[index];
        offset += 2048; index ++; }
ptr += ratio;
if (ptr > loopend) {ptr -= looplength;}                      // (optional loop control)
```

1.5 Virtual Analog (VA) Oscillators

VA oscillators are intended to emulate the analog originals with focus on classic waveforms and excellent modulation capabilities. The designs presented here generate high fidelity signals in both frequency and time domain using an efficient algorithm that furthermore supports easy integration of FM and hard synchronization.

We start with the bandlimited impulse train (BLIT) as described in [1]. The following designs are significant improvements as they do not suffer from any of the commonly encountered problems like frequency dependent gain, bias or phase shift.

The first signal to be synthesized is a simple sawtooth, which can be obtained by integrating the sum of an impulse train and a constant (Fig. 15).

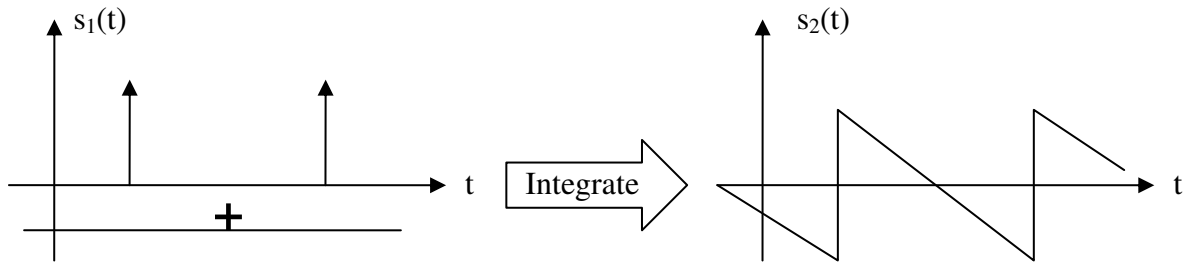


Fig. 15: Simple Sawtooth by Integrating an Ideal Impulse Train

Earlier insights from Fig. 3 suggest using bandlimited impulses to prevent aliasing when the signal is translated to discrete time as it is the case in every digital oscillator (Fig. 16). If the impulse has a flat spectrum over the audio band, no difference is perceived. An eventual high frequency drop can be compensated by applying a filter to the output of the oscillator.

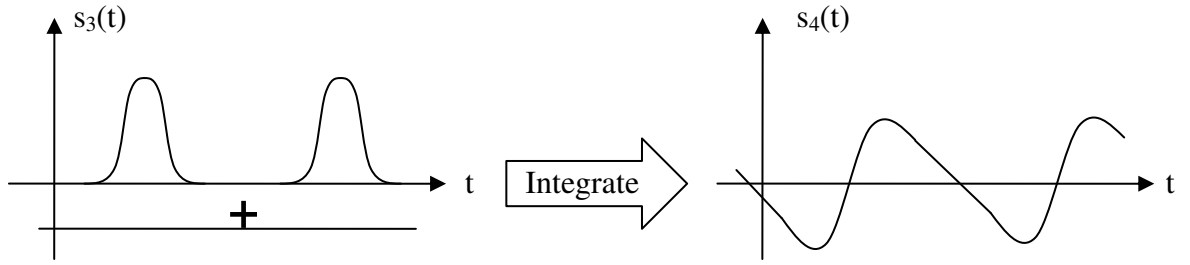
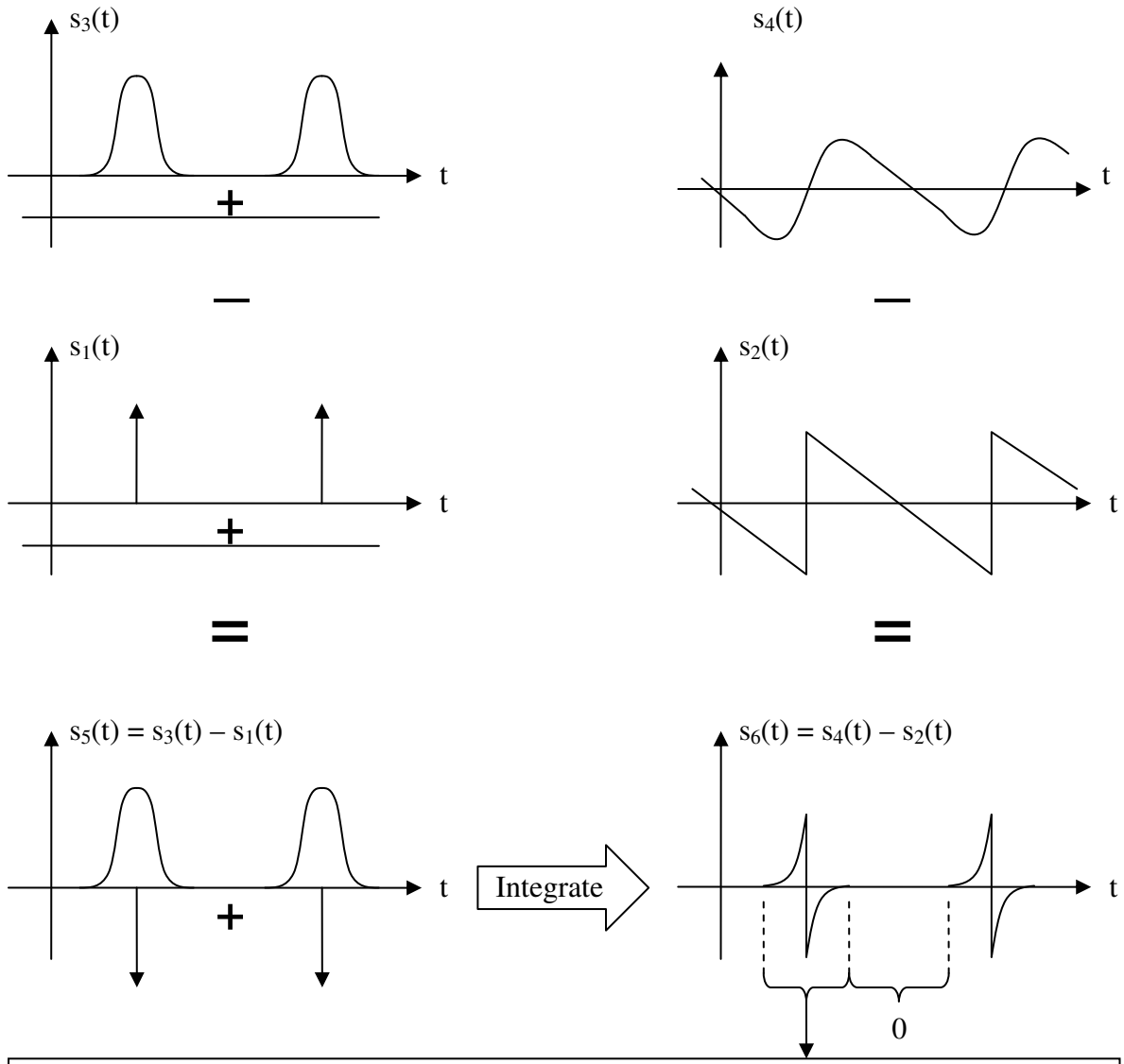


Fig. 16: Bandlimited Sawtooth by Integrating a Bandlimited Impulse Train

In a practical implementation, an open integrator would be very undesirable due to its frequency response and numerical properties. A nice trick similar to the BLEP approach [5] gets rid of it (Fig. 17):

Integrate the difference of a bandlimited and an ideal impulse train. This removes the constant part and leads to a fixed length signal segment ready for tabulation. At runtime, read out the segment and add a simple sawtooth.



Constant length segment! Calculated once from $s_5(t)$ and stored in a table.

Runtime Operations:

1. Compute $s_2(t)$ and $s_6(t)$ using the generic oscillator structure of Fig. 1
2. Compute the output $s_4(t) = s_2(t) + s_6(t)$

Fig. 17: Bandlimited Sawtooth Generation without Integration

According to Fig. 17, the signal s_6 is a train of fixed length segments separated by zero-valued regions whose length depends on the fundamental frequency. An upper limit for the fundamental is imposed by the length of the bandlimited impulse, being shortest in the design of Fig. 6. In order to create the segment, we calculate the cumulative sum of s_5. This simple approximation to continuous time integration produces a negligible high frequency gain for practical table sizes.

Now we are up to determine the appropriate table size for the segment. The output signal only partially consists of tabulated values. That's why Eq. 1 cannot be applied directly, but a conservative guess can be made treating the whole signal as a table.

Since a sawtooth has a 1/f spectrum and aliasing is proportional to f, every harmonic contributes to the same amount. The spacing of the harmonics is preserved when they are aliased, thus only one component can fall more than octave below f_o and we apply Eq.1 to it. Components above f_o remain inaudible because they are masked. In order to keep aliasing below -85 dB and assuming a fundamental frequency $f_o = 4$ kHz and a sample rate $f_s = 48$ kHz the number of table entries becomes $N \approx 2700$ per sampling interval T. The entire segment spans 4T, due to symmetry we just have to store $N \approx 5400$ for one half.

The bandlimited sawtooth inherits the spectral droop from the bandlimited impulse. To retrieve the brilliance of the analog original, we insert a postfilter, which should be as simple as possible because it consumes processing time. A suitable IIR filter is found readily via cut and try:

$$H_{pf}(z) = \frac{1}{0.65 + 0.35z^{-1}}$$

Figures 18 to 20 show the oscillator's spectral purity and temporal fidelity. We happily notice how little aliasing falls below the fundamental when the signal is discretized in time. Refer to appendix B3 for the Matlab code used to create the segment.

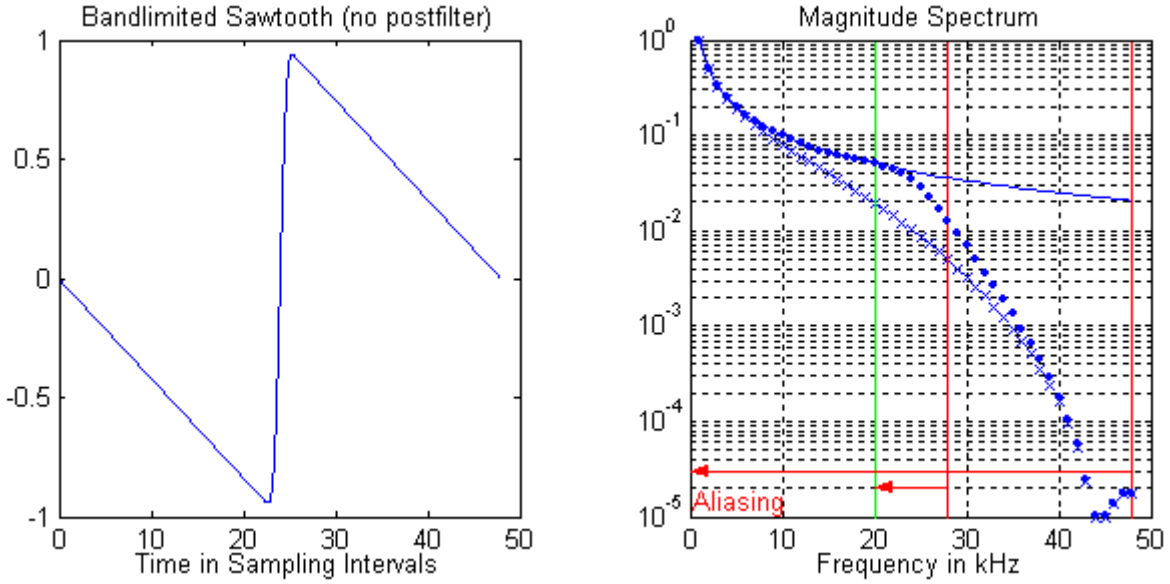


Fig. 18: Bandlimited Sawtooth, f_o = 1 kHz
(Cross: No postfilter. Dot: With postfilter. Line: Theoretical sawtooth spectral envelope.)

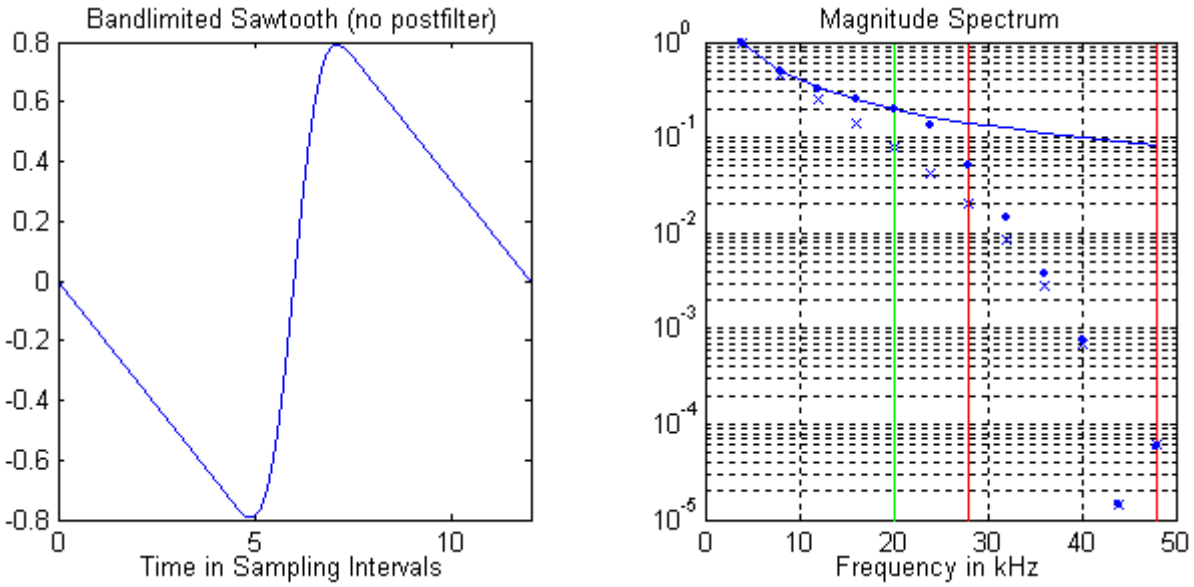


Fig. 19: Bandlimited Sawtooth, f_o = 4 kHz

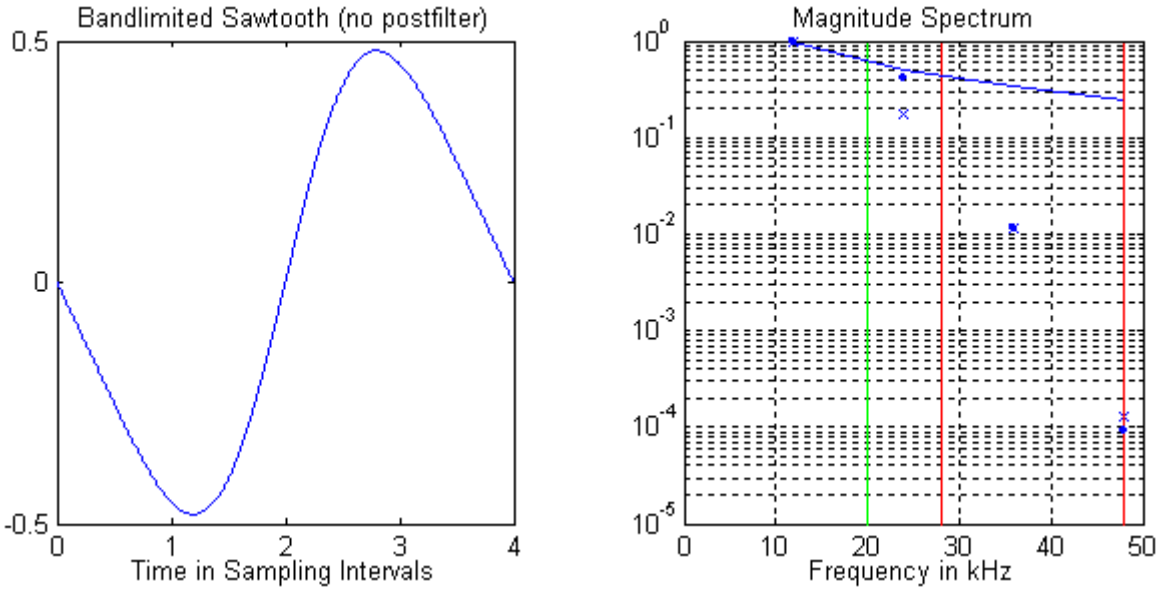


Fig. 20: Bandlimited Sawtooth, f_o = 12 kHz

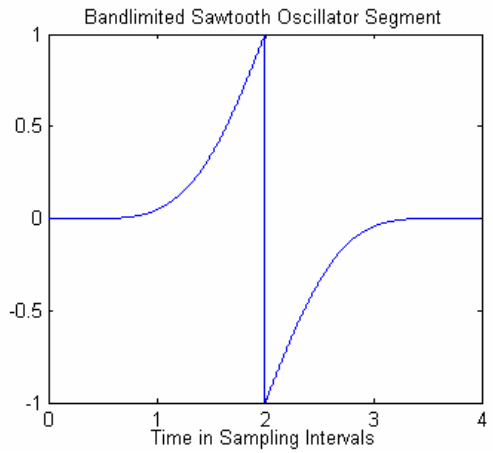


Fig. 21: Segment Function

The actual sawtooth oscillator is derived from the generic oscillator in Fig. 1. It employs a constant 90° phase shift to centre the segment at the zero crossing. Fig. 22 and 23 depict the block diagram and associated signals. Many synthesizers mix several oscillator signals together. In this case, the postfilter is required only once. It's a good idea to place it after a ring modulator or a similar nonlinearity to lower their tendency to produce aliasing.

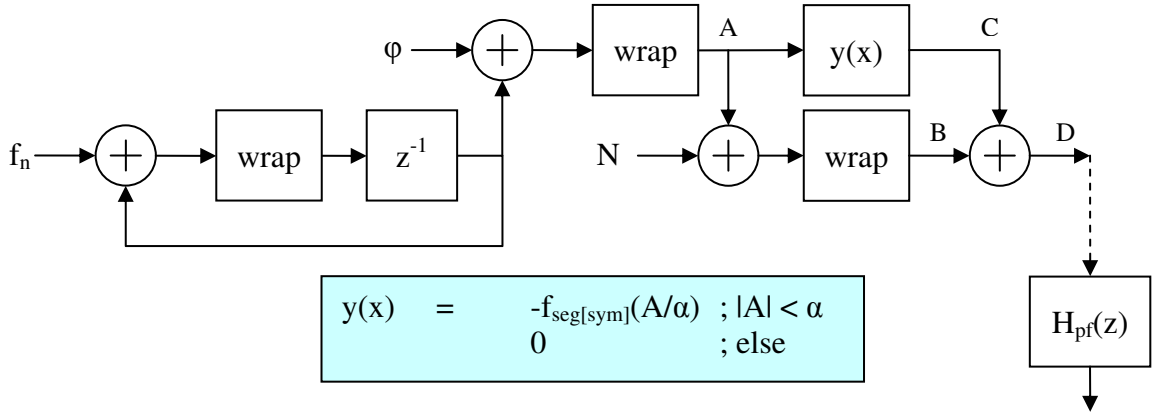


Fig. 22: Bandlimited Sawtooth Oscillator

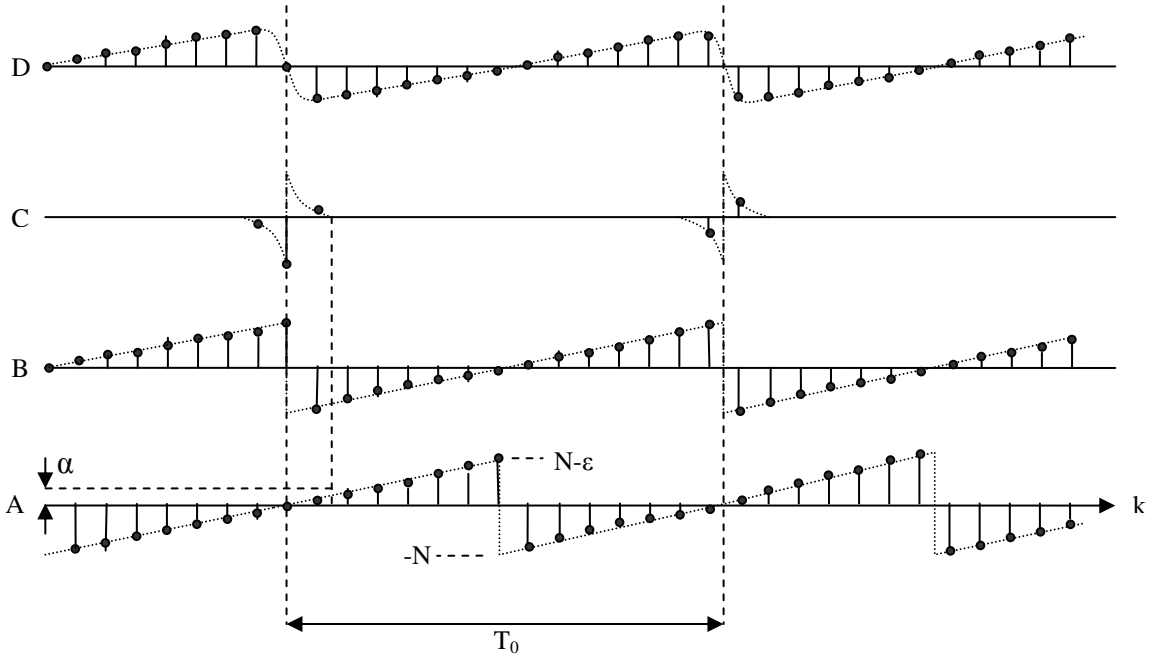


Fig. 23: Signals in the Bandlimited Sawtooth Oscillator

Signal A completes a cycle within T_o ranging through 2N. As the half segment has a length of 2T, we obtain $\alpha = 4NT/T_o = f_o \cdot [4N/f_s]$. The index i to read the segment function from a table containing M entries per half becomes $i = MA/\alpha = A \cdot [1/f_o] \cdot [Mf_s/(4N)]$. The brackets show which products are constant and which ones must be computed during runtime. We observe that the fundamental frequency f_o is required in its reciprocal form too. Since synthesizers apply an exponential characteristic to frequency control and there's no need to update at audio rate (FM acts on the phase modulation input φ), the additional effort is not obstructive (see appendix C2).

In case the hardware does not provide enough memory, the half segment can be approximated by a polynomial (Fig. 24). The high order is typical for non-periodic bandlimited functions.

$-f_{seg[half]}(x) = \sum_{n=0}^{7} c_n x^n$	$c_0 = 0.99986$ $c_1 = -2.97566$ $c_2 = -0.23930$ $c_3 = 7.83529$ $c_4 = -3.25094$ $c_5 = -11.51283$ $c_6 = 13.50376$ $c_7 = -4.36023$	(Polynomial Approximation)

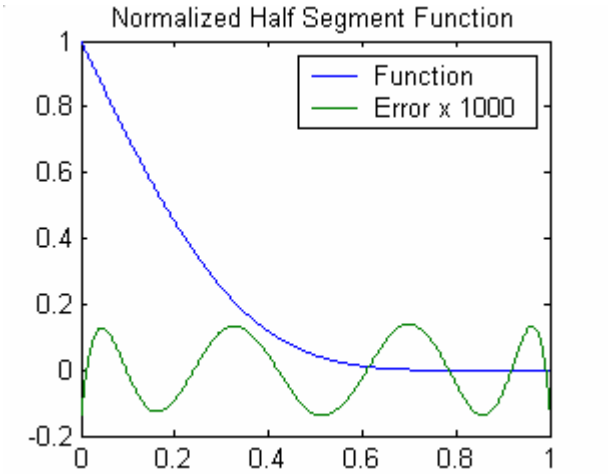


Fig. 24: Polynomial Approximation of the Half Segment

Although the same segment could be used to generate a bandlimited square wave, pulse width modulation (PWM) is often desirable. If we sum the output of two sawtooth oscillators running in opposite direction, we get a variable width pulse wave that inherits zero bias and spectral purity (Fig. 25). The width is proportional to an offset φ_{pw} added to the phase modulation input of one oscillator. A square wave is obtained for $\varphi_{pw} = N/2$ (90°).

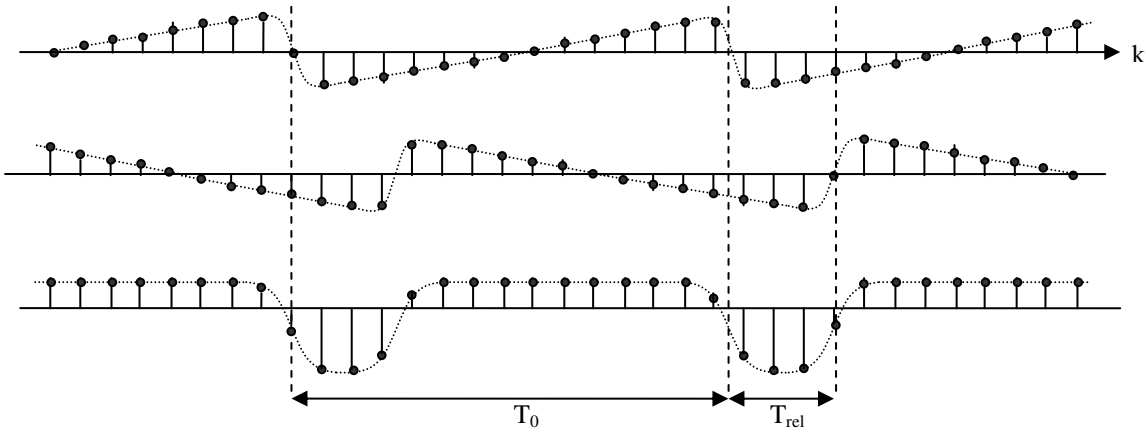


Fig. 25: Variable Width Bandlimited Pulse Generation

A triangle wave is synthesized as the time integral of a square wave by combining the ideas of Fig. 17 and Fig. 25. See Fig. 26 for details. The segment in s_{10} is calculated as the cumulative sum of the sawtooth oscillator segment and then tabulated.

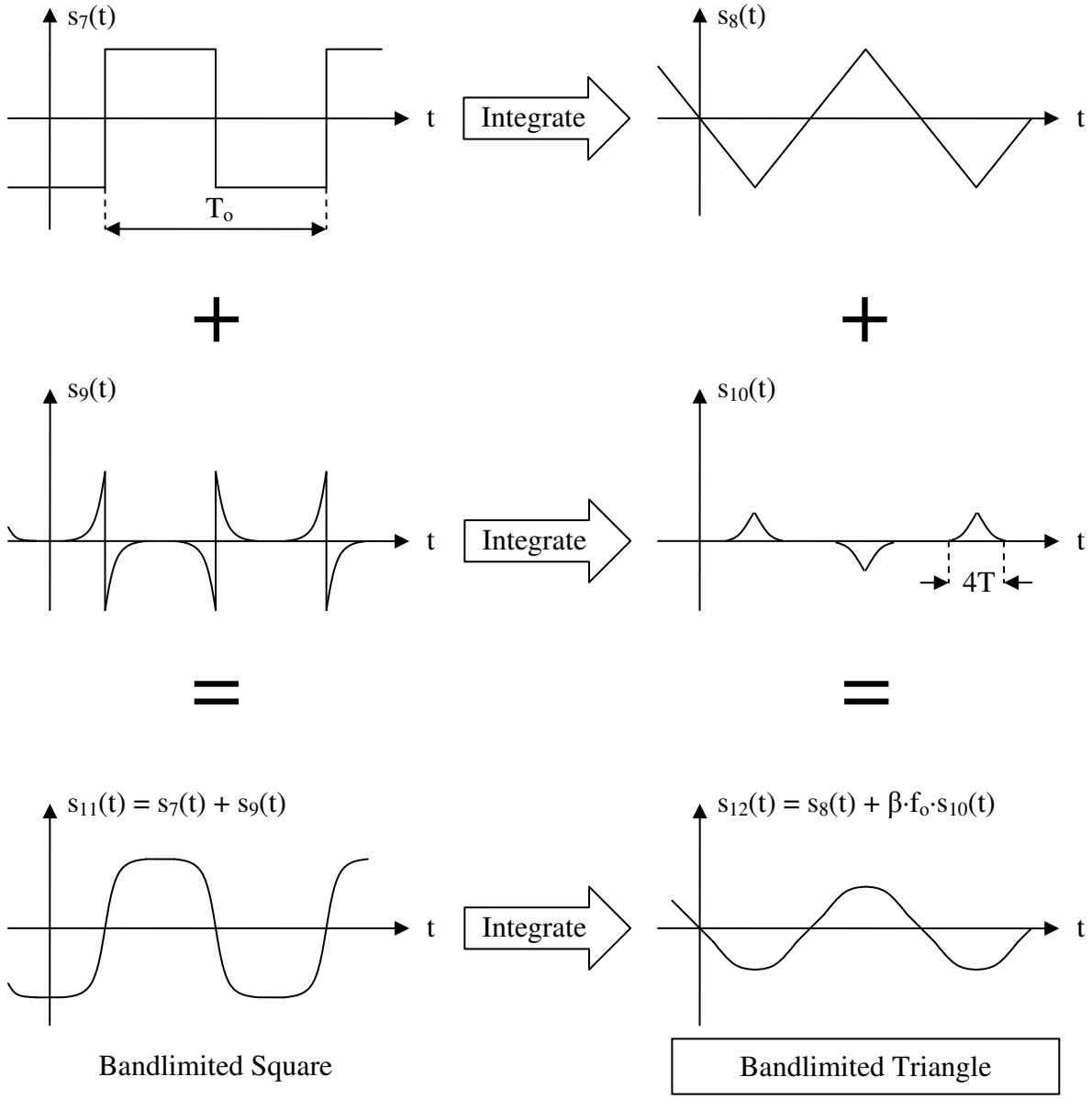


Fig. 26: Bandlimited Triangle Generation

As s_8 is the time integral of s_7, its amplitude is proportional to the fundamental period T_o. On the other hand, a segment of s_9 has constant size and amplitude which results in constant amplitude and size of s_{10}. To keep the amplitude of s_{12} independent of the fundamental frequency f_o, we use a constant amplitude signal s_8 and scale s_{10} inversely proportional to T_o. This is equivalent to weight it proportionally to f_o. The constant β is found by matching the slopes of s_8 and s_{10} at an arbitrary f_o and may be condensed into the tabulated values.

The weighted addition relaxes the accuracy requirements for the signal s_{10}. Further examination reveals that the table size can be reduced to $N \approx 1000$ for the whole segment. Unfortunately, we need two segments per full period and they overlap for $T_o < 8T$, hence a wide range triangle oscillator contains two segment generators (Fig. 27) making it as resource hungry as a pulse oscillator. Unlike most hardware synthesizers, which use dedicated constant resources for an oscillator, software synthesizers running on a general purpose computer consume processing cycles on demand. In this case, we should consider replacing the triangle with a sinusoidal oscillator for $f > f_s/8$ to save a segment generator.

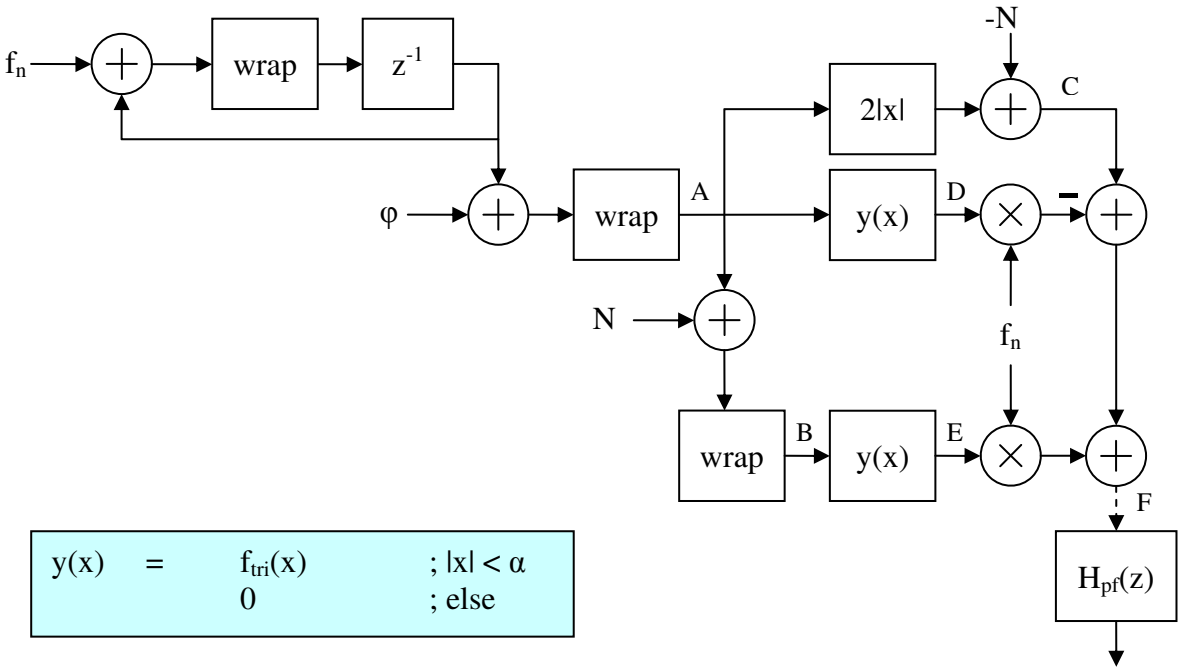


Fig. 27: Wide Range Bandlimited Triangle Oscillator

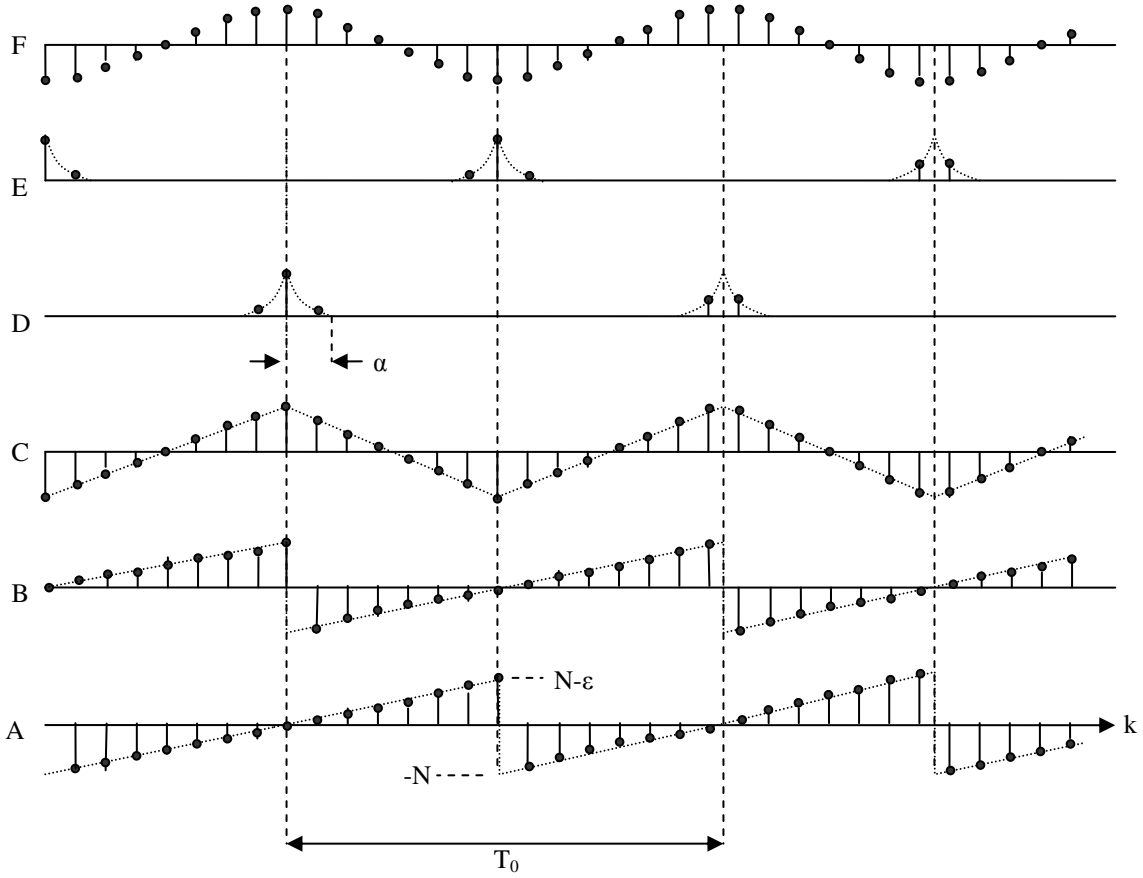


Fig. 28: Signals in the Bandlimited Triangle Oscillator

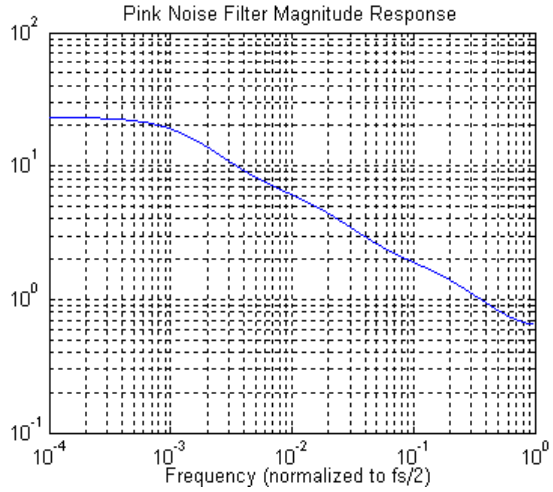
Analogous to the sawtooth oscillator, we obtain $\alpha = f_o \cdot [4N/f_s]$. The index to read the segment function from a table containing M entries per half becomes $i = A \cdot [1/f_o] \cdot [Mf_s/(4N)]$.

1.6 Noise Generation

White noise for musical purposes can be synthesized efficiently by means of a linear congruential generator [2]. Modulo division is performed automatically as a wrap-around if the divisor matches the word size.

For 32 bit integer data types: $x = (69069x + 1) \bmod 2^{32}$
On 24 bit fixed point audio DSPs: $x = (12268885x + 1) \bmod 2^{24}$

Pink noise with its -10dB/decade slope and constant power per octave is obtained by filtering white noise. A filter with 0.3 dB deviation from $0.00045f_s$ to $0.45f_s$ has been proposed by Robert Bristow-Johnson:



$$H(z) = \frac{(z - 0.984436)(z - 0.833923)(z - 0.075684)}{(z - 0.995728)(z - 0.947906)(z - 0.535675)}$$

Fig. 29: Pink Noise Filter

This filter converts white noise uniformly distributed from -1 to 1 to non-uniformly distributed pink noise with RMS amplitude 1. As this implies both an overall gain of 4.77 dB and an increased crest factor, subsequent stages should be able to process peak values of at least 3 without excessive distortion.

1.7 Sinusoidal Oscillators

There are many ways to generate a sine wave, the most efficient being 2nd order closed-loop systems based on trigonometric identities or the z-transform of the discrete-time sinusoid (section 2.1). Despite their low computational demand when running free, obstructive calculations are mandatory for transient-less arbitrary modulation of both frequency and phase. On the other hand, fast sinusoidal oscillators with immediate linear phase control are readily realized with the generic oscillator structure of section 1.1.

If the hardware supports fast integer arithmetic and memory access, a lookup table is usually the preferred choice to convert the simple sawtooth to a sinusoidal. About 500 table entries are sufficient with linear interpolation, nearest neighbour lookup is even faster at the cost of a larger table - about 30000 entries for a high-quality full cycle. Furthermore, we should be aware of some advantages of integer arithmetic: Sums automatically wrap around on overflow, and a simple shift operation converts the phase accumulator to the table index.

Polynomial approximations are a good alternative when multiplication is fast but memory access expensive. Although it's often suggested that reduction theorems should be used to narrow the input range to $\pi/2$ or $\pi/4$, this is mainly an issue with truncated Taylor series. Since conditional instructions take more time than an additional product term evaluation on modern processors, we will stick to a polynomial that maps the whole circle and can be fed directly with the output of a simple sawtooth oscillator. Minimax or Chebyshev techniques spread the error evenly over the entire range outperforming Taylor series by several orders of magnitude in both time and spectral domain. It's important to note that the error consists of harmonics rather than random noise: If we avoid discontinuities at the output when the input wraps from 1 to -1 or vice versa, higher harmonics will decline quickly with increasing order and even an approximation with relatively large error may sound good as long as aliased harmonics remain inaudible. This idea is exploited in the designs of Fig. 30 to 34.

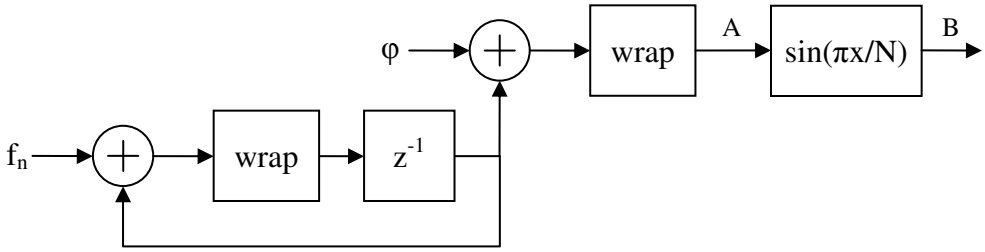


Fig. 30: Map-Based Sinusoidal Oscillator

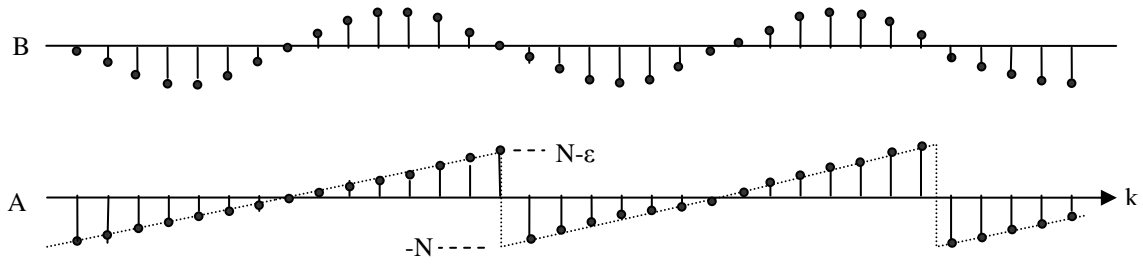


Fig. 31: Signals in the Map-Based Sinusoidal Oscillator

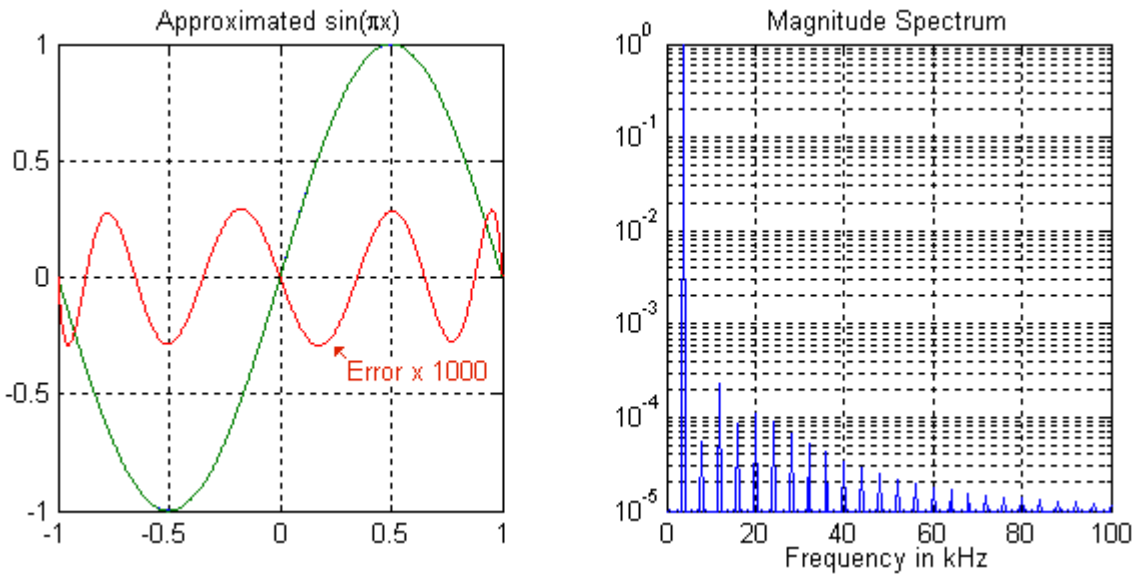


Fig. 32: $\sin(\pi x) \approx -0.433645x^7 + 2.428288x^5 - 5.133625x^3 + 3.138982x$

The function in Fig. 32 works well as a general purpose sine shaper for musical applications. All harmonics are masked and aliased components falling below a fundamental of 4 kHz are 90 dB down for a sampling rate f_s = 48 kHz. Fig. 33 and 34 show the performance of the next higher and lower order polynomial. For comparison, see the truncated Taylor series in Fig. 35.

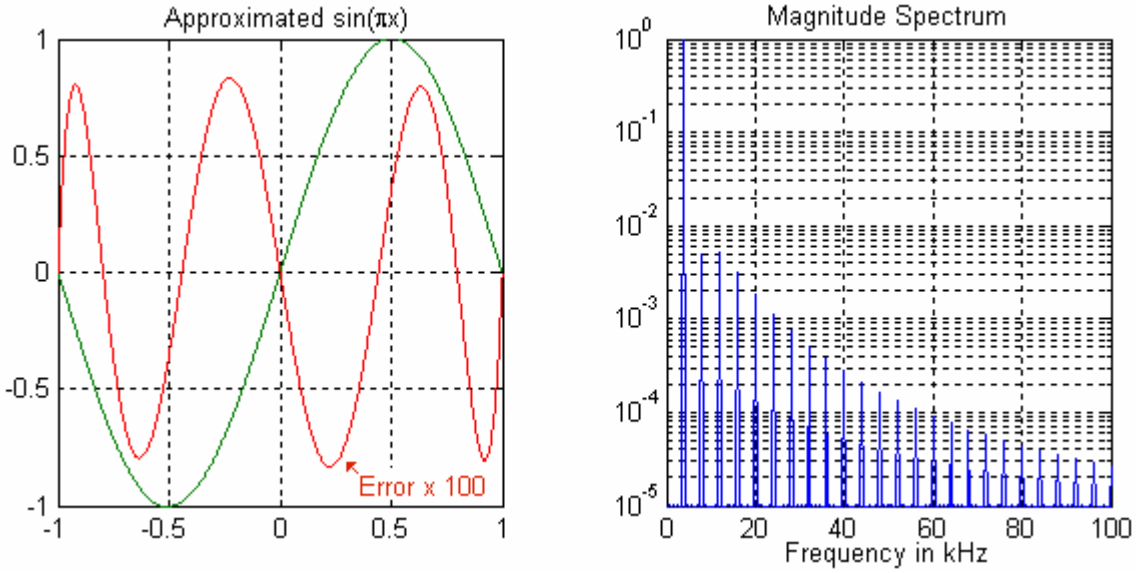


Fig. 33: $\sin(\pi x) \approx 1.63190x^5 - 4.71594x^3 + 3.08404x$

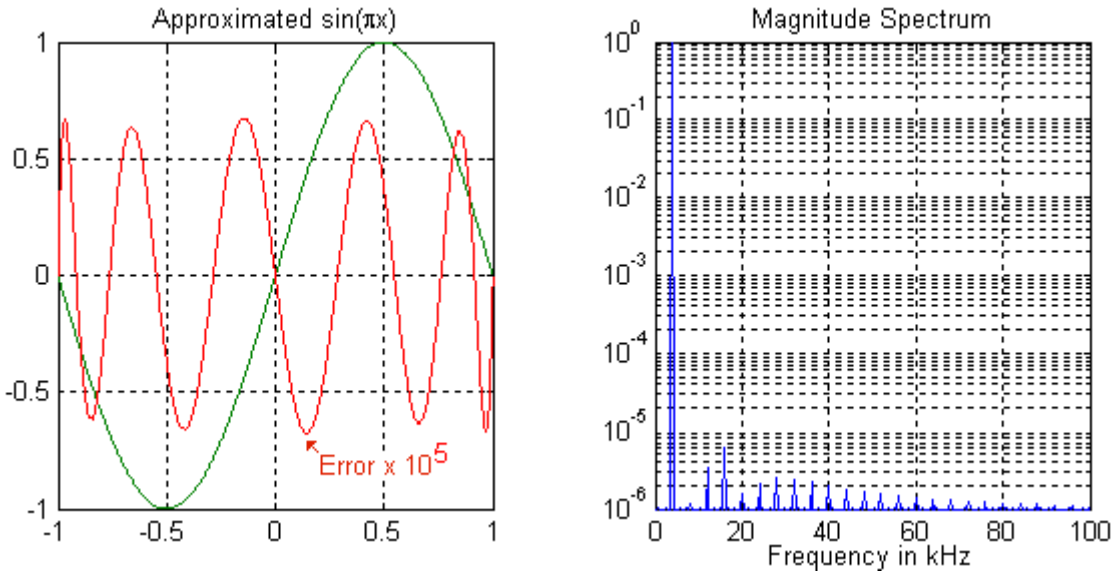


Fig. 34: $\sin(\pi x) \approx 0.0636716x^9 - 0.5811243x^7 + 2.5422065x^5 - 5.1662729x^3 + 3.1415191x$

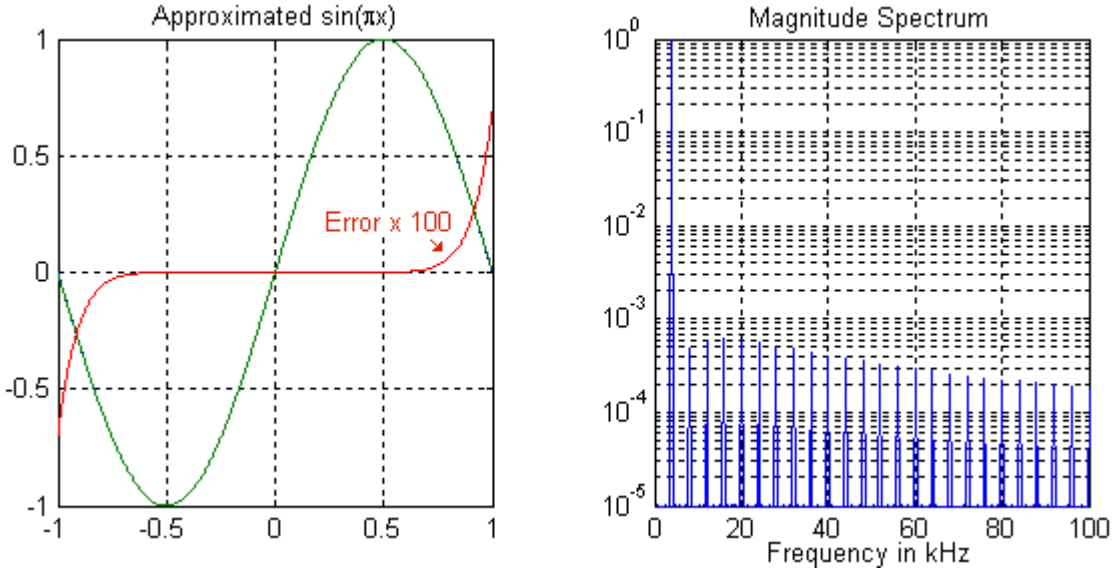


Fig. 35: Truncated Taylor Series, $\sin(\pi x) \approx (\pi x)^9/9! - (\pi x)^7/7! + (\pi x)^5/5! - (\pi x)^3/3! + \pi x$

Approximations of the cosine within the range [-π, π] are:

ε_{max} [%]	K_2	K_3	K_4	K_5	K_6	$P(x) \approx \cos(\pi x)$
6.3	-24	-39	-49	-53	-58	$2.0124x^4 - 4.0060x^2 + 0.9339$
3.0	-32	-43	-42	-45	-47	$2.4236x^4 - 4.3650x^2 + 0.9693$
0.25	-59	-54	-62	-69	-75	$-0.8775x^6 + 3.7472x^4 - 4.8648x^2 + 0.9975$
0.004	-108	-94	-91	-96	-107	$0.17824x^8 - 1.28739x^6 + 4.04196x^4 - 4.93273\,x^2 + 0.99996$

Fig. 36: Performance of Selected Sawtooth to Cosine Shaper Polynomials
($K_n = A(nf_o)/A(f_o)$ in dB)

Sometimes, a triangle is available. In this case, one product term can be saved since the approximation only needs to cover [-π/2, π/2].

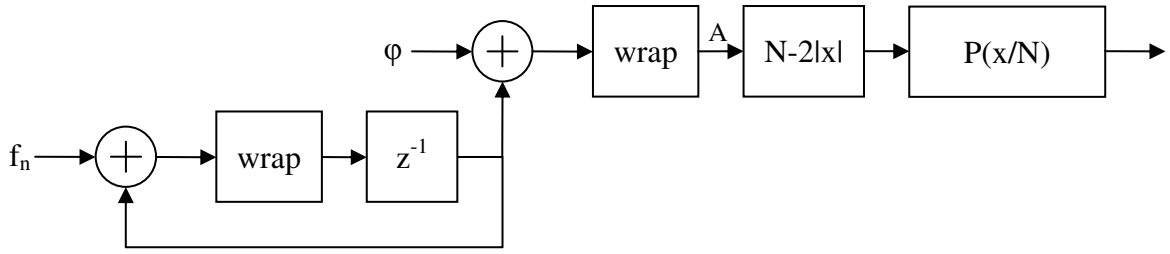


Fig. 37: Mapped-Triangle Sinusoidal Oscillator

ε_{max} [%]	K_3	K_5	K_7	K_9	K_{11}	$P(x) \approx \sin(\pi x/2)$
1.21	-38	-57	-70	-80	-91	$1.5209x - 0.5090x^3$
0.46	-49	-55	-57	-61	-64	$1.5478x - 0.5520x^3$
0.011	-91	-80	-91	-102	-117	$1.57007x - 0.64089x^3 + 0.070726x^5$
0.007	-98	-84	-120	-98	-98	$1.57031x - 0.64209x^3 + 0.071844x^5$
0.00006	-147	-132	-126	-135	-153	$1.5707908x - 0.6458911x^3 + 0.0794309x^5 - 0.0043311x^7$

Fig. 38: Performance of Selected Triangle to Sine Shaper Polynomials
($K_n = A(nf_o)/A(f_o)$ in dB, K = 0 for n even)

The output is a cosine wave. Hence, we could attach a saw-to-sine shaper after the wrapper (node A) to build a quadrature oscillator.

1.8 Ring Modulation

In the synthesis stage of musical instruments amplitude modulation at audio rates is usually accomplished in a ring modulator, which does nothing but multiply two input signals to form the output. Typical applications are the generation of disharmonic timbres, additional harmonics, and tuned noise. A multiplication in the time domain corresponds to a convolution of the spectra: If one input has a component at the frequency f_1, the other one at f_2, then the output will contain components at $|f_1 \pm f_2|$. New frequencies are created, so we have to take care of aliasing and keep in mind that oscillator signals extend beyond the audio band (Fig. 18). Fig. 39 shows some signals for typical conditions at f_s = 48 kHz with two components.

Input 1:	Sine f_1 = 1.1 kHz, A = 0.9	+	Sine f_{e1} = 23.1 kHz, A = 0.1
Input 2:	Sine f_2 = 5.5 kHz, A = 0.9	+	Sine f_{e2} = 22.0 kHz, A = 0.1

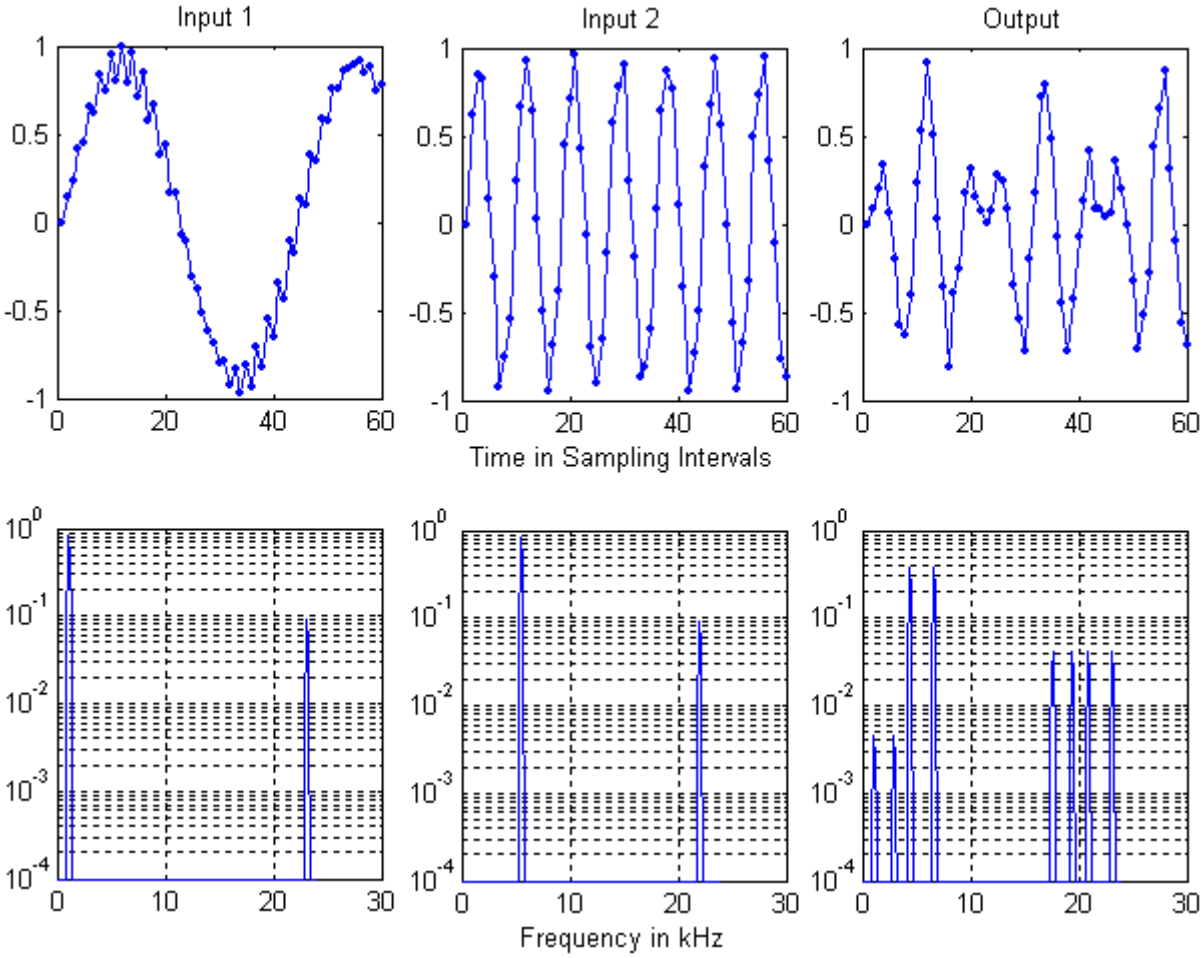


Fig. 39: Signals in the Simple Ring Modulator

An analysis yields:

1. The desired components at $f_1 \pm f_2$ = 4.4 and 6.6 kHz.
2. Components from a strong desired low frequency signal and an out-of-band signal $f_1 \pm f_{e2}$ = 23.1 and 20.9 kHz, $f_2 \pm f_{e1}$ = 28.6 kHz (aliased to 19.4 kHz) and 16.5 kHz. Not objectionable or masked in practical cases.
3. Components from two weak top- or out-of-band signals. $f_{e1} \pm f_{e2}$ = 45.1 kHz (aliased to 2.9 kHz) and 1.1 kHz. Audible. Aliased components are especially problematic because they are neither masked nor do they fit into the harmonic context.

Conclusion: Audible aliasing mostly arises from two input components around $f_s/2$. It's a good idea to eliminate them in a filter with a zero at Nyquist before they get multiplied.

The collateral high frequency attenuation is equalized by a filter after the multiplication. We should also insert a DC trap (s. Appendix C1) since multiplying signals with commensurate frequency components leads to a (usually small) constant bias that may disturb subsequent stages. (Fig. 40 and 41)

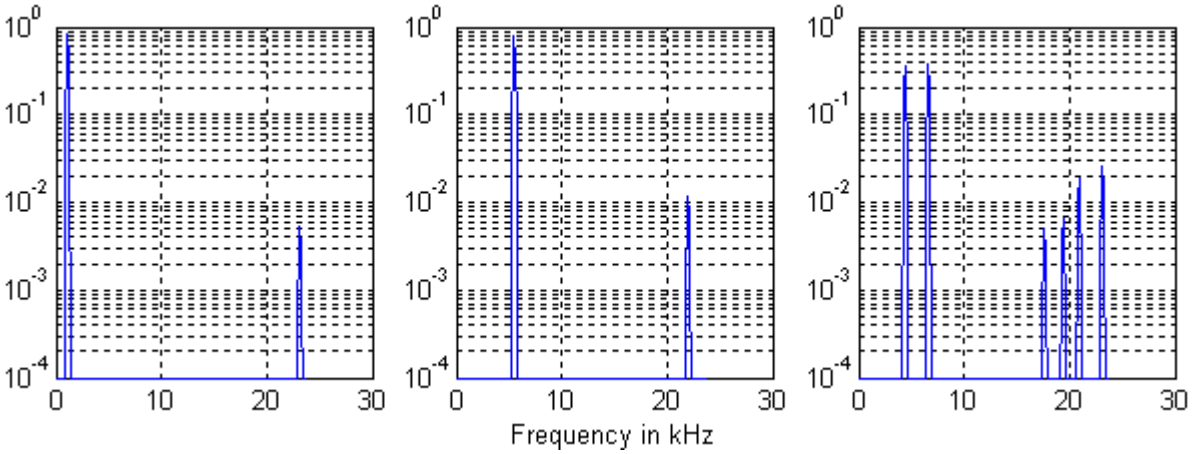


Fig. 40: Spectra of the Enhanced Ring Modulator (at points A, B, C)

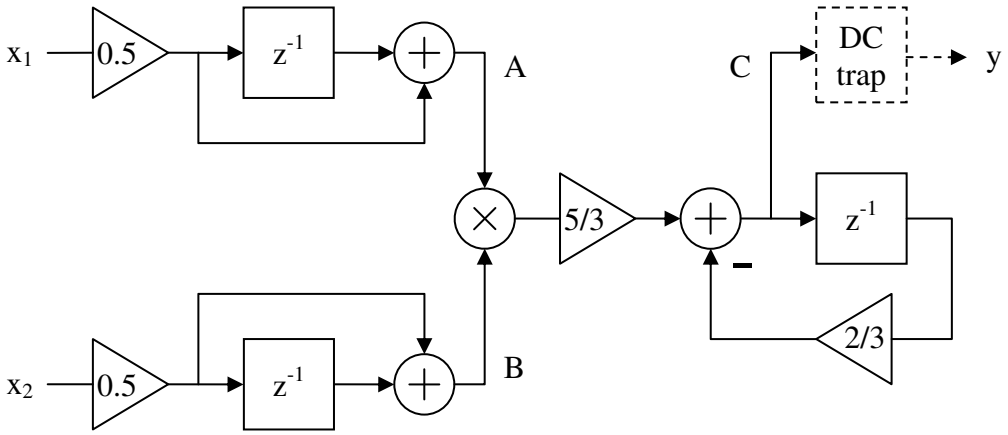


Fig. 41: Enhanced Ring Modulator

Input scaling by 0.5 prevents overflow at nodes A and B. In floating point systems, we may omit it and replace the factor 5/3 by 5/12.

1.9 FM Synthesis

1.9.1 Principles

Frequency modulation (FM) is a popular sound synthesis method introduced in [13]. It generates inherently bandlimited complex spectra with low computational effort. Most of today's musical synthesizers feature some sort of FM, often modulating the phase instead of the frequency. Phase modulation (PM) is actually used in classic FM synthesizers; thus we stick to the term "FM synthesis" but name the modulation type correctly. We start by examining the general behaviour of a basic setup (Fig. 42).

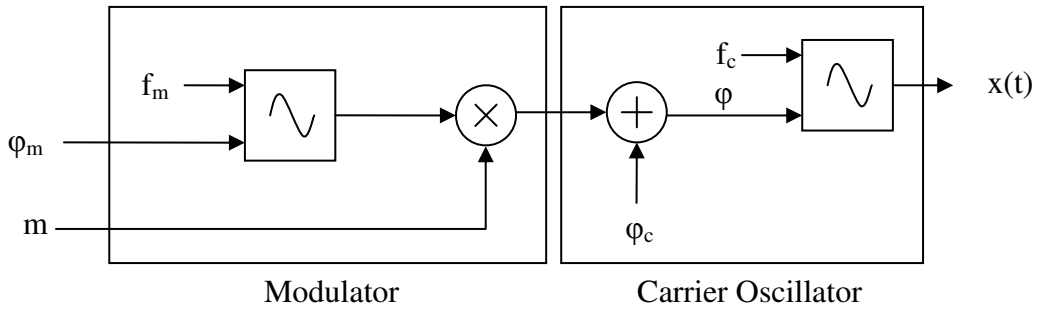


Fig. 42: Basic Phase Modulation Setup (as used in FM synthesis)

Key formulae governing the system of Fig. 42 are:

$$x(t) = A\sin\left(2\pi f_c t + m\sin\left(2\pi f_m t + \varphi_m\right) + \varphi_c\right) = A\sum_{n=-\infty}^{\infty} J_n(m)\sin\left(2\pi\left(f_c + nf_m\right)t + n\varphi_m + \varphi_c\right)$$

Discussion:

1. Frequency components appear at $|f_c \pm nf_m|$ with n integer. For a harmonic output spectrum, the modulator frequency f_m must be an integer multiple or submultiple of the carrier frequency f_c.

2. The magnitude of spectral components is determined by n-th order Bessel functions of the modulation index m: A larger m results in a richer harmonic content. For n > m, the harmonics will diminish rapidly effectively limiting the output spectrum (Fig. 43). With the Stirling formula for the factorial it can be shown that this function decreases slightly faster than exponentially with increasing n.

3. Frequency components can appear twice with different magnitude and opposite phase causing characteristic holes in the spectrum. Signals with very high m tend to sound annoying due to the characteristic peak around mf_m; values above 10 are rarely used. Furthermore, the harmonics do not evolve naturally when m is swept in order to create a dynamic spectrum. In this case it's advisable to confine m to about 1.5 and use a more complex modulator or feedback.

4. If $\varphi_c = \varphi_m = 0$, the output will be unbiased. This also holds for $\varphi_m \neq 0$ if φ_m is the output of an additional modulator with $\varphi_c = 0$ and φ_m is either zero or satisfying the condition in this sentence. If $\varphi_c = 0$ for all oscillators in a setup, the output is unbiased. As this is often hard to fulfil in a variable synthesis chain, the frequency control input is rarely used for modulation to avoid detuning. A bias at the phase input only changes the spectrum and often goes unnoticed at all.

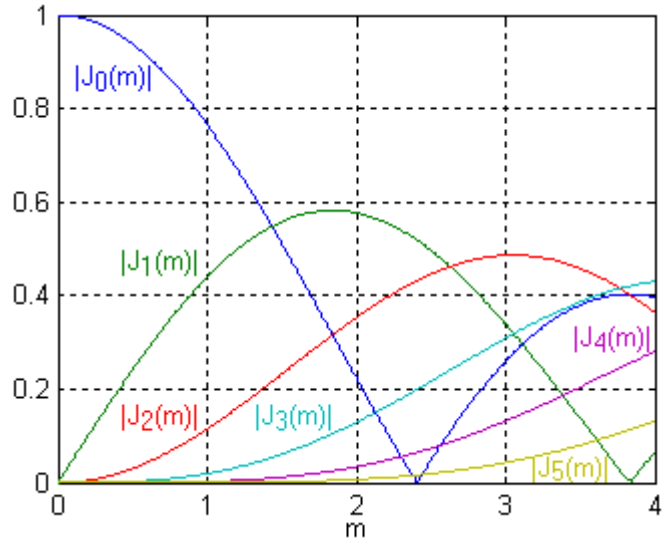


Fig. 43: Magnitude of Bessel Functions of the First Kind

There's an asymptotic estimate valid for $0 < m < \sqrt{n+1}$: $J_n(m) \approx \frac{1}{n!}\left(\frac{m}{2}\right)^n$

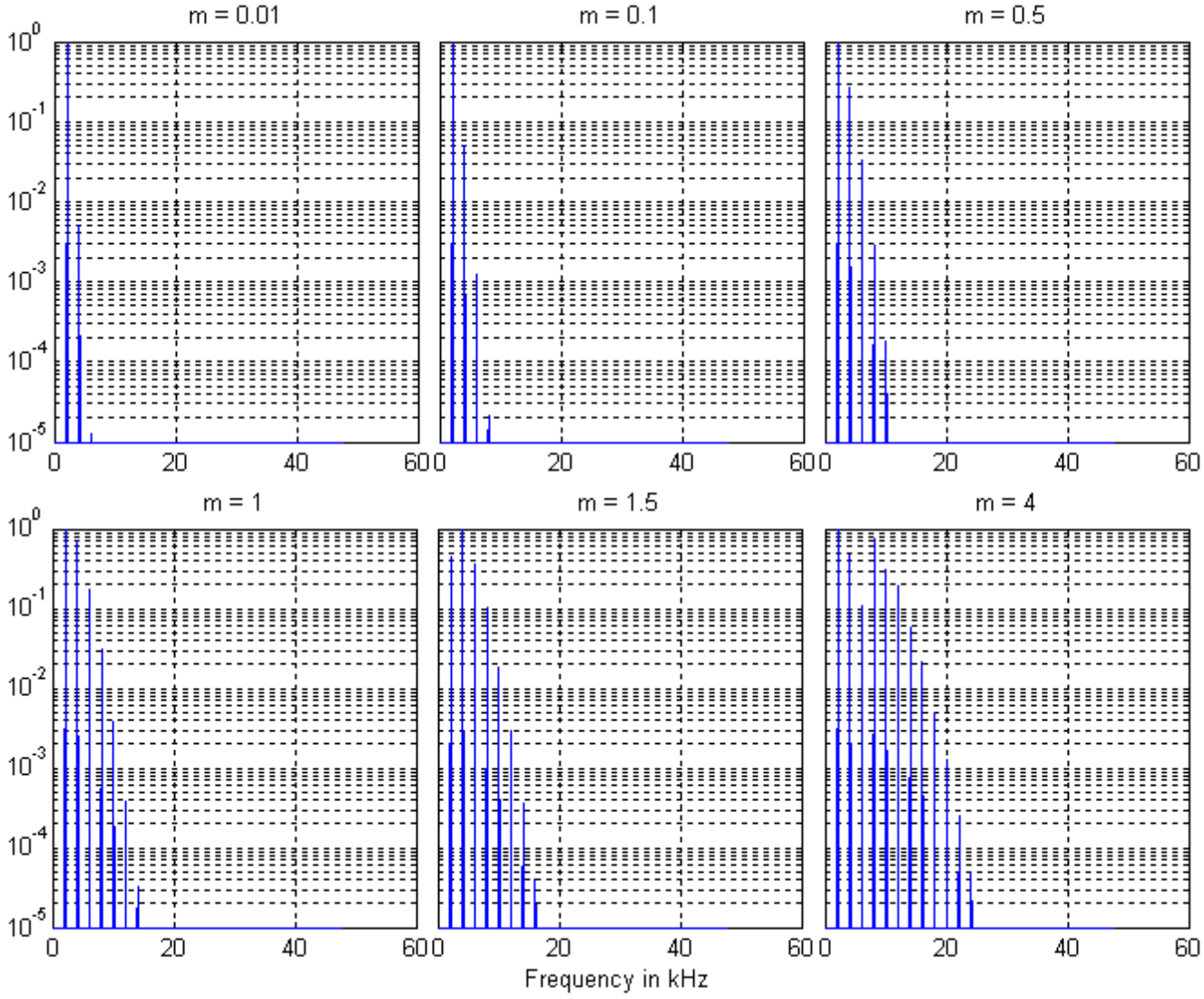


Fig. 44: Basic Setup Spectra ($f_m = f_c = 2$ kHz, $\varphi_m = \varphi_c = 0$)

Fig. 44 shows FM spectra for different values of m. Here we see the remarkably steep roll-off that substantially simplifies bandwidth control. A provision for adjusting the modulation index m according to the fundamental frequency suffices to keep an FM system alias-free. The frequency f_{max} at which the components are down by at least 80 dB relative to the main peak is bounded by:

$$f_{\max} < f_c + 2(m+1.3)f_m \qquad ; \text{m} > 0.03$$

$$f_{\max} < f_c - \frac{9.3}{\ln(0.5m)} f_m \qquad ; 0.0002 < \text{m} < 0.03$$

A severe disadvantage of the basic setup is the lack of highs. Three methods are commonly employed as a remedy:

- Choosing $f_m >> f_c$
- Feedback
- Modulating the modulator

Fig. 45 depicts the result of the first approach with its characteristic large holes in the spectrum. This technique is often applied to create metallic timbres and short transients.

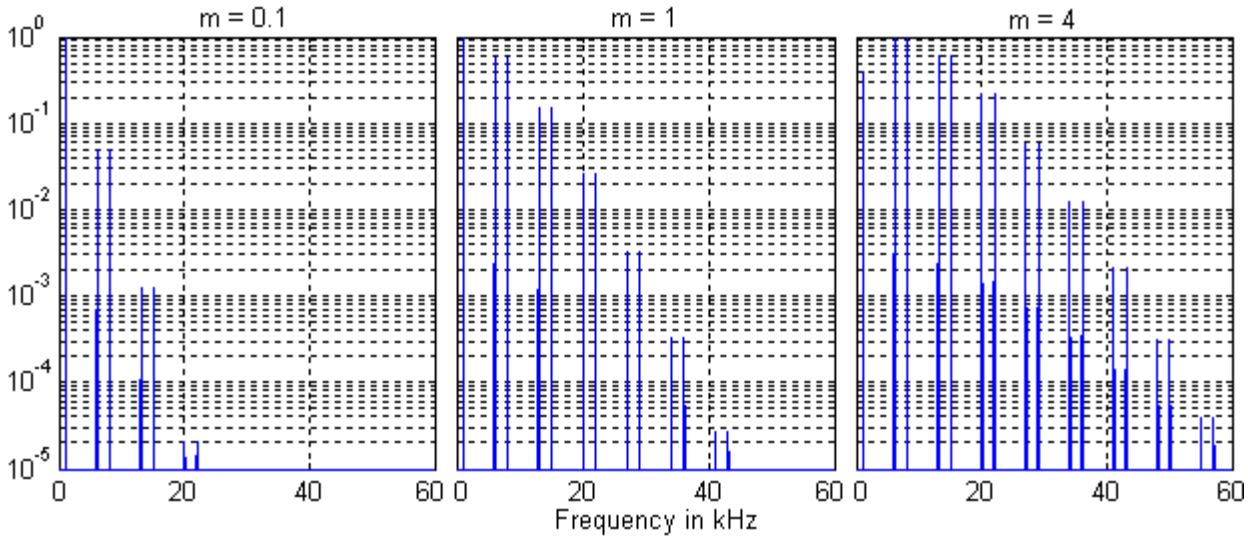


Fig. 45: Basic Setup Spectra (f_m = 7 kHz, f_c =1 kHz, $\varphi_m = \varphi_c = 0$)

1.9.2 Feedback

The spectral characteristics and control dynamics of the feedback method resemble those of subtractive synthesis. Hence, feedback pleasantly extends the sonic range of FM synthesis towards analog timbres.

So far, we assumed the system to be time continuous. If we use the phase input as we did, the modulation paths are memoryless and the discrete time realization exactly equals the sampled continuous system. Therefore, we just have to take care for the continuous prototype not to contain spectral components that will cause audible fold-over. A discrete time feedback system however must have some memory in its loop and neglecting it will not provide a viable approximation beyond $f_s/10$. That's why it's advisable to directly analyze a discrete time version (Fig. 46) that employs a sinusoidal oscillator from section 1.7 (e.g. Fig. 30). For further analysis, the phase input φ is normalized to 2π and the output amplitude to 1.

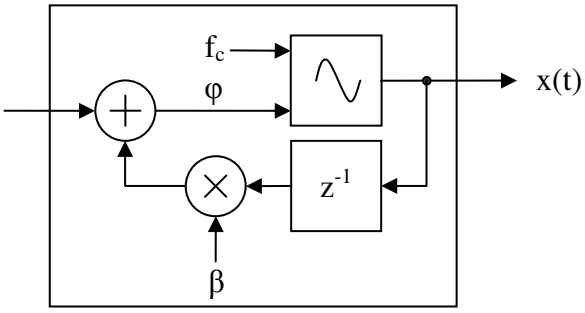


Fig. 46: Discrete Time Oscillator with PM Feedback

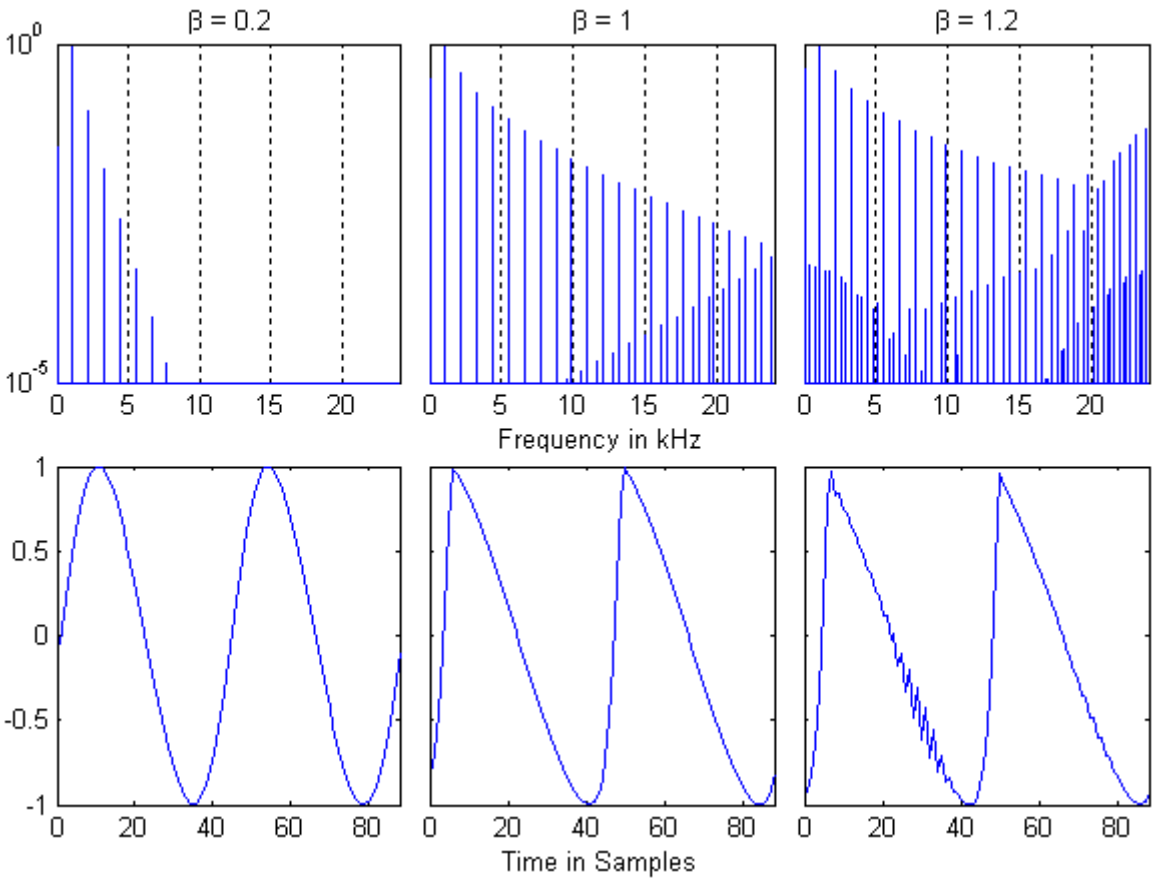


Fig. 47: PM Feedback Spectra

Typical spectra are shown in Fig. 47. The harmonics grow smoothly without leaving holes when β is increased. Unfortunately, the richness of a sawtooth cannot be achieved as the system becomes unstable for β > 1. We also notice the fold-over for larger values of β. The spectral envelope for different values of β in Fig. 48 may assist in determining whether aliasing would become problematic with the intended amount of feedback.

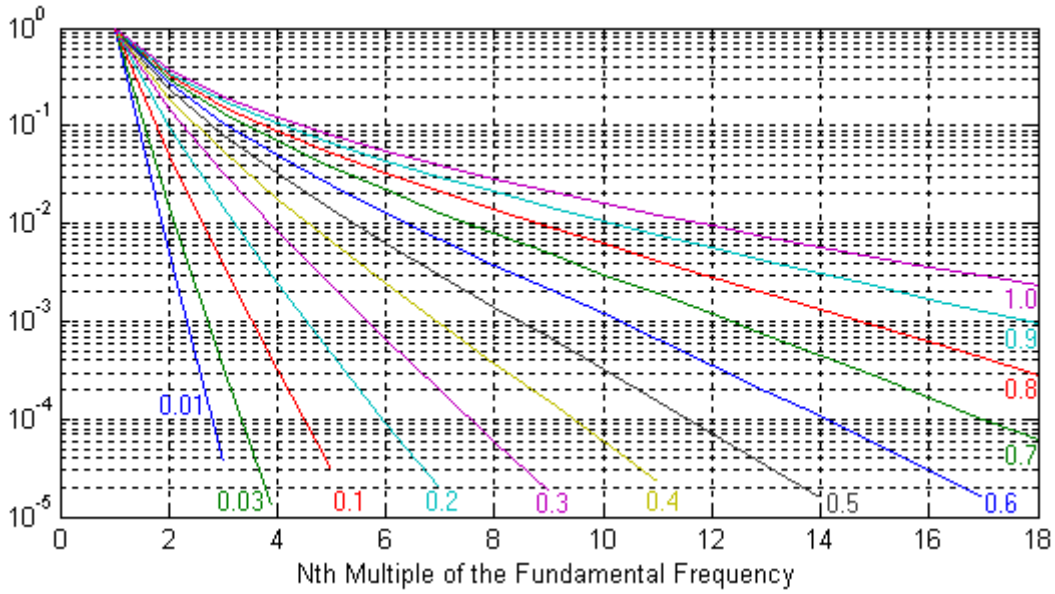


Fig. 48: PM Feedback Normalized Spectral Envelopes for different values of β

The amplitudes in Fig. 48 are approximately (n > 1): $A_n(\beta) \approx \left(0.3 + 1.48e^{-n}\right)\left(\beta - 0.25\beta^3\right)^{n-1}$

A side effect of PM feedback is a frequency dependent output bias, roughly given for $f_o/f_s < 0.1$ and practical β by:

$$x_{bias} \approx -\left(3.12\beta + 2.69\beta^3\right)\frac{f_0}{f_s}$$

Consequently, a DC trap (s. Appendix C1) is recommended at the summation point of the carrier oscillators if feedback is employed.

In classical FM synthesizers, the instability at β >> 1 is exploited to create noise. It sounds nearly white except for some extra peaks related to the fundamental, which may be desired or not.

Examination of the instability reveals that it starts with a high frequency parasitic oscillation peaking at $f_s/2$. So why not reject those components in the feedback loop to extend the useful range of β and get a more brilliant sound? Not surprisingly we recycle the technique already known from the ring modulator. The resulting structure dates back to the early days of FM although it has rarely been mentioned in the literature. Recently, a variation based on an IIR low-pass filter has been proposed by Peter Schoffhauzer. (Fig. 49)

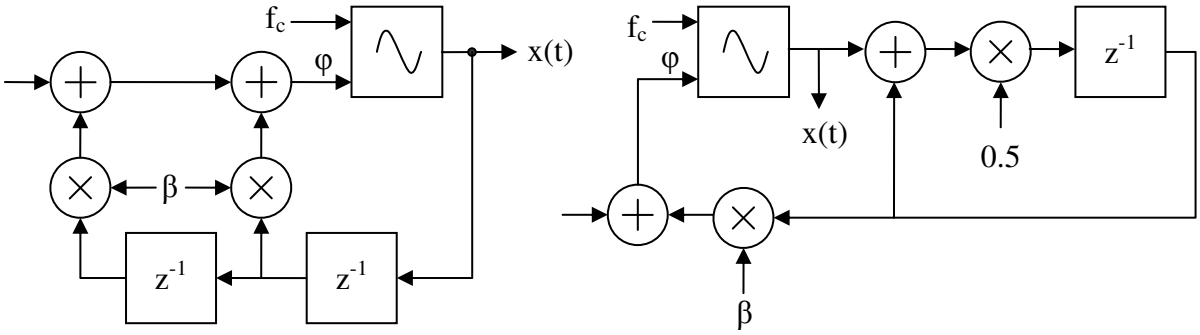


Fig. 49: Enhanced PM Feedback (Left: Traditional, Right: Schoffhauzer)

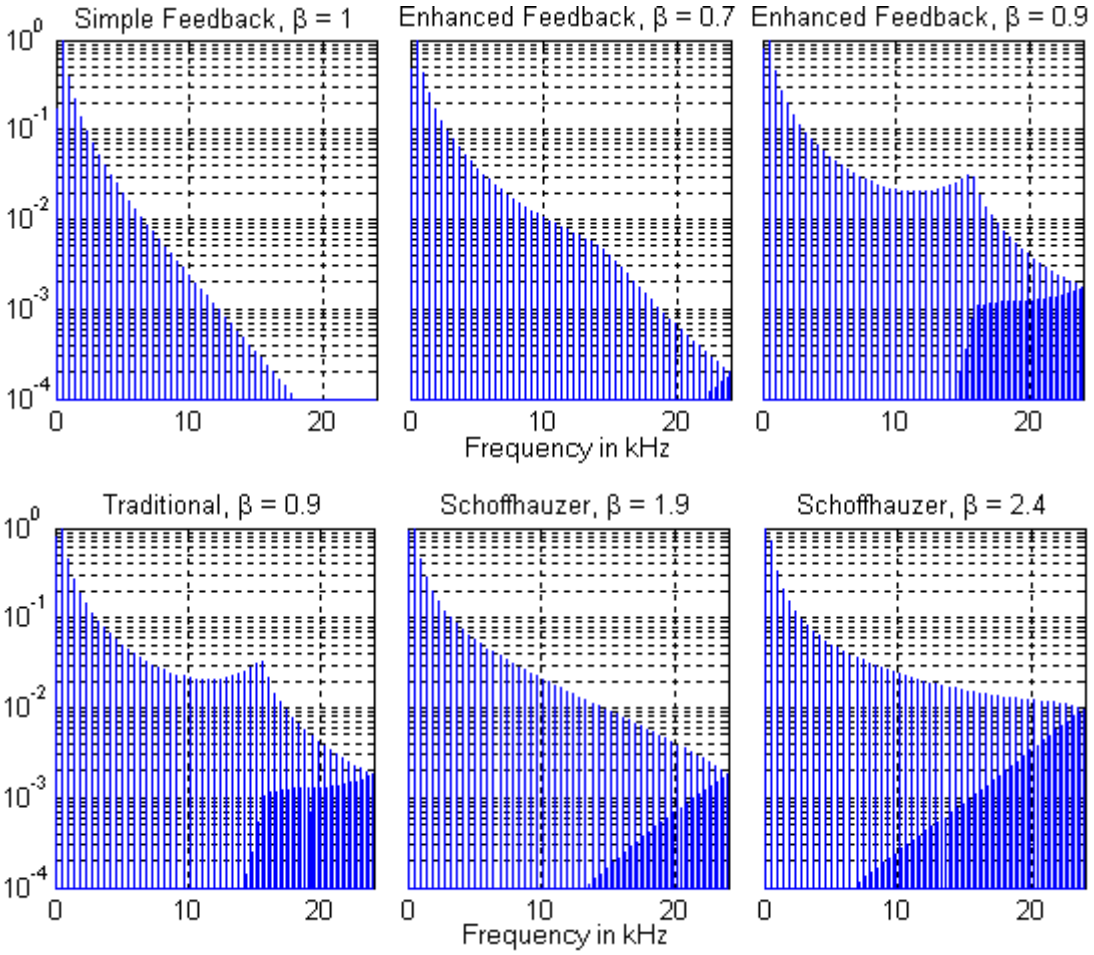


Fig. 50: Comparison of PM Feedback Spectra (f_0 = 450 Hz)

Enhanced feedback results in a distinctly richer spectrum, but the traditional method introduces a peak around $f_s/3$ that becomes more pronounced with increasing β. While it's an amazing improvement over simple feedback in any FM system, we should refrain from emulating an analog sawtooth using this structure, lest we end up with an annoying peak that shines through in every sound. The Schoffhauzer method performs much better in this regard and a good VA sawtooth spectrum is obtained with a few additional tweaks. Refer to his paper for details [16]. Note the moved output tap in Fig. 49.

1.9.3 Complex Modulators

Modulators with complex spectra are very rewarding and ubiquitous in FM synthesis. Complex spectra are obtained by modulating a modulator or using non-sinusoidal oscillators. The static spectrum of such a modulator should sound rather dull to avoid aliasing as the following calculations demonstrate. We consider a modulator signal that consists of a strong low frequency (A_{m1}) and a weak high frequency (A_{m2}) component. Furthermore, we introduce angular frequencies ($\omega = 2\pi f$) for compact notation:

$$x_m(t) = A_{m1}\sin(\omega_{m1}t) + A_{m2}\sin(\omega_{m2}t)$$

The modulated carrier output becomes:

$$x(t) = \sin\left(\omega_c t + m\left[A_{m1}\sin(\omega_{m1}t) + A_{m2}\sin(\omega_{m2}t)\right]\right)$$
$$x(t) = \sin\left(\omega_c t + mA_{m1}\sin(\omega_{m1}t)\right)\cos\left(mA_{m2}\sin(\omega_{m2}t)\right) + \cos\left(\omega_c t + mA_{m1}\sin(\omega_{m1}t)\right)\sin\left(mA_{m2}\sin(\omega_{m2}t)\right)$$

With $mA_{m2} << 1$ and 2nd order Taylor approximations for sine and cosine:

$$x(t) \approx \sin\left(\omega_c t + mA_{m1}\sin(\omega_{m1}t)\right)\left[1 - \frac{m^2 {A_{m2}}^2}{2}\sin^2(\omega_{m2}t)\right] + \cos\left(\omega_c t + mA_{m1}\sin(\omega_{m1}t)\right)mA_{m2}\sin(\omega_{m2}t)$$
$$x(t) \approx \sin\left(\omega_c t + mA_{m1}\sin(\omega_{m1}t)\right)\left[1 - \frac{m^2 {A_{m2}}^2}{4}\left(1 - \cos(2\omega_{m2}t)\right)\right] + \cos\left(\omega_c t + mA_{m1}\sin(\omega_{m1}t)\right)mA_{m2}\sin(\omega_{m2}t)$$

$$x(t) \approx \left[1 - \frac{m^2 {A_{m2}}^2}{4}\right]\sin\left(\omega_c t + mA_{m1}\sin(\omega_{m1}t)\right) +$$
$$\frac{m^2 {A_{m2}}^2}{8}\left[\sin\left((\omega_c + 2\omega_{m2})t + mA_{m1}\sin(\omega_{m1}t)\right) + \sin\left((\omega_c - 2\omega_{m2})t + mA_{m1}\sin(\omega_{m1}t)\right)\right] +$$
$$\frac{mA_{m2}}{2}\left[\sin\left((\omega_c + \omega_{m2})t + mA_{m1}\sin(\omega_{m1}t)\right) - \sin\left((\omega_c - \omega_{m2})t + mA_{m1}\sin(\omega_{m1}t)\right)\right]$$

Discussion:

1. The main component is the carrier modulated by the strong signal.

2. There are two weak copies of the main component, spectrally shifted by the weak signal's frequency. If both main and weak components extend beyond the audio range, audible aliasing becomes likely. This problem is encountered for example when FM is applied to VA oscillators.

3. Even weaker spectral copies of the main component occur shifted by multiples of the weak signal's frequency (here we calculated only 2nd order components). Despite their low amplitude, they are often more troublesome than those mentioned in (2), because aliasing may also crop up with a low frequency sinusoidal carrier. This is the main reason why dull modulators are preferable.

4. The output is unbiased if all oscillators are in phase and unbiased.

See Fig. 51 for an example.

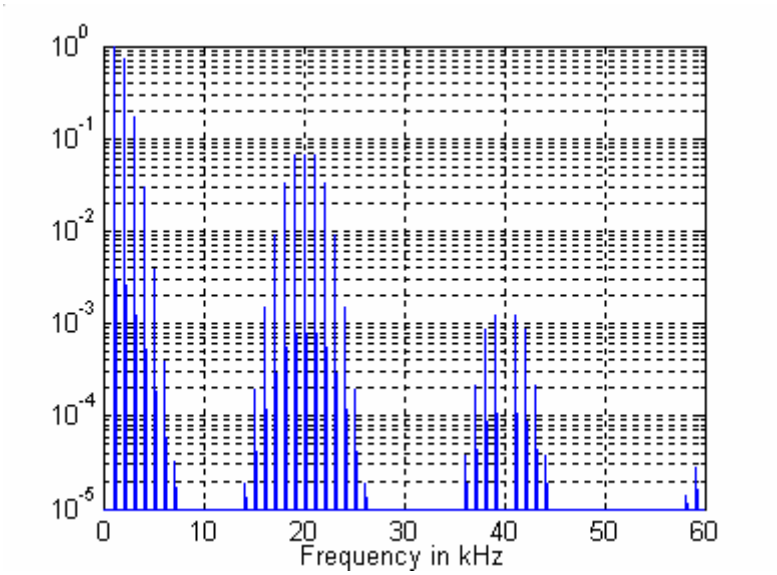


Fig. 51: PM Spectrum, Modulator: Sum of Sines (1 kHz, m = 1) + (20 kHz, m = 0.1)

To illustrate the combination of the foregoing methods, we aim at replacing a FM feedback pair for a VA sawtooth oscillator (Fig. 52). Let's see what we get in a straightforward approach: A clean spectrum with an interesting grumpy character caused by the shape becoming exponential at low frequencies similar to some analog counterparts (Fig. 53). We may easily add extensions to morph from a sawtooth to a sinusoid and for phase modulation. The downside: It sounds less brilliant for very low fundamentals (enhanced feedback will improve). Due to feedback, the system lacks immediate phase control; frequency control is somewhat cumbersome too. The simplistic phase compensation is imperfect and leaves the output slightly biased; we may use a look-up table or omit it in favour of a DC trap. All in all, this oscillator is only recommended if there is already a given classic FM infrastructure.

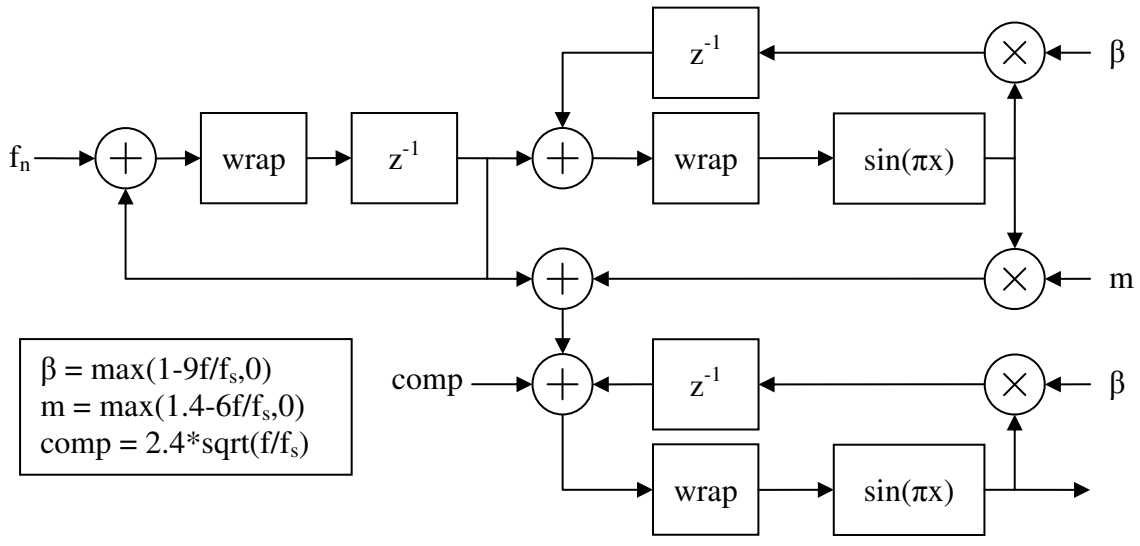


Fig. 52: PM Bandlimited Sawtooth Oscillator

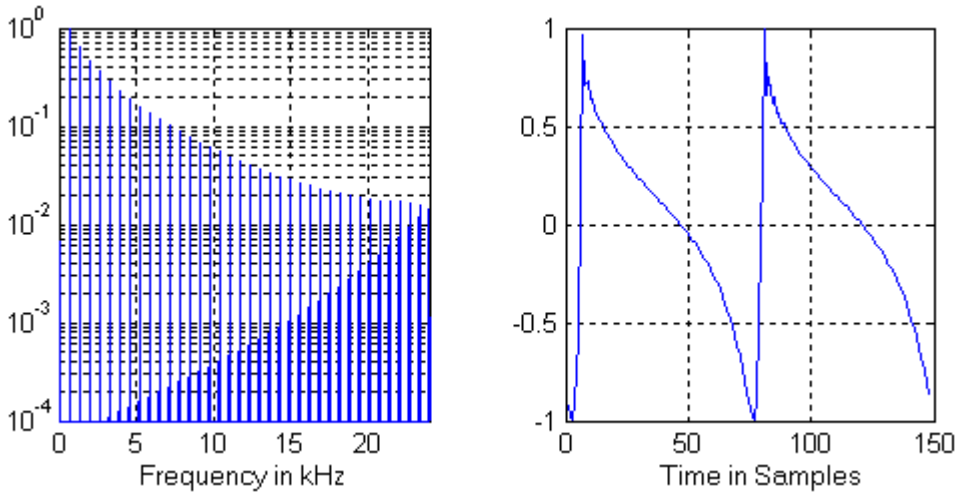
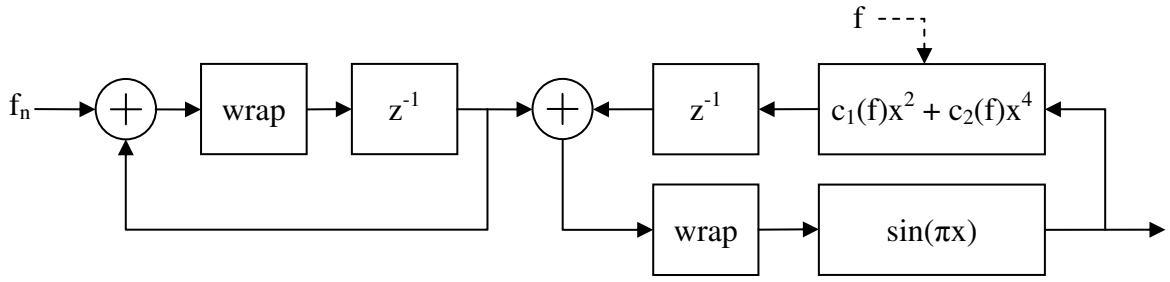


Fig. 53: Spectrum and Shape of the PM Bandlimited Sawtooth Oscillator (f_o = 650 Hz)

In a second example, we focus on the triangle which is likely to be modelled more accurately due to its sparse spectrum. Such a replacement would be very beneficial since the VA triangle oscillator is computationally intensive. In addition, we come across the idea of inserting a polynomial into the feedback path. An even order type is chosen as the triangle spectrum contains only odd harmonics.



$c_1 = (0.75 - c_2) \cdot \min(\max(1.44 - 11.5f/f_s, 0), 1)$
$c_2 = \max(0.5 - 12f/f_s, 0)$ (can be set to zero with minor sonic change)

Fig. 54: PM Bandlimited Triangle Oscillator

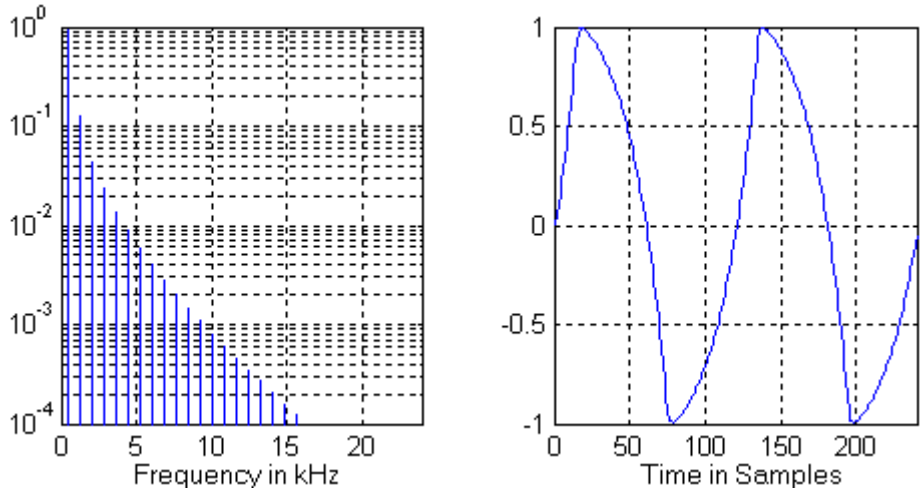


Fig. 55: Spectrum and Shape of the PM Bandlimited Triangle Oscillator (f_o = 400 Hz)

In the triangle oscillator, spectral accuracy is not limited by the maximum amount of feedback. Instead, excessive feedback will evoke an unnatural boost of the lower harmonics.

To summarize the reasonable effort FM approach to classic analog waveforms, one can state that a clean and well-controlled spectrum is obtained. The lack of immediate phase control as well as weak highs at very low fundamentals may be obstructive in some cases.

We finish the section tackling the challenge of adding FM to VA oscillators. Already known is that a modulator should have as little high frequency content as possible to avoid aliasing. Does this also apply to the carrier? To analyze it, we assume without loss of generality an unbiased carrier decomposed into a sine-based Fourier series:

$$x_c(t) = \sum_{n=1}^{\infty} b_n \sin(n\omega_c t + \varphi_n)$$

A phase shift $\Delta\varphi$ at the modulation input just shifts the time pointer of the waveform cycle by $\Delta t = T\Delta\varphi/(2\pi) = \Delta\varphi/\omega_c$. Hence, the modulated carrier becomes:

$$x_{c(\mathrm{mod})}(t) = \sum_{n=1}^{\infty} b_n \sin(n\omega_c\left[t + \frac{\Delta\varphi}{\omega_c}\right] + \varphi_n) = \sum_{n=1}^{\infty} b_n \sin(n\omega_c t + \varphi_n + n\Delta\varphi)$$

Assuming further a modulator $\Delta\varphi = m\cdot\sin(\omega_m t)$ we obtain:

$$x_{c(\mathrm{mod})}(t) = \sum_{n=1}^{\infty} b_n \sin(n\omega_c t + \varphi_n + nm\sin(\omega_m t)) = \sum_{n=1}^{\infty} b_n \sum_{k=-\infty}^{\infty} J_k(nm)\sin(\left[n\omega_c + k\omega_m\right]t + \varphi_n)$$

The Bessel function $J_k(nm)$ starts to diminish for $k > nm$. Given n_{max} as the order of the highest significant carrier harmonic, phase modulation causes a bandwidth increase roughly proportional to $n_{max}mf_m$. To get rid of the frequency dependence, which may easily cause aliasing, we make the effective m inversely proportional to f_m by weighting the modulator spectrum. The simplest way would be to integrate the modulating signal. This is feasible at no cost using the frequency instead of the phase control input at the carrier oscillator, because phase is the time integral of frequency. However, any bias in the modulator would detune the carrier oscillator and the modulation depth becomes excessively large for low fundamentals. In some systems, the modulator is unbiased by design and true frequency modulation may be adequate. Otherwise, the setup in Fig. 56 is recommended: The phase input is fed by a leaky integrator with an additional zero at $f_s/2$. Exact values are application specific.

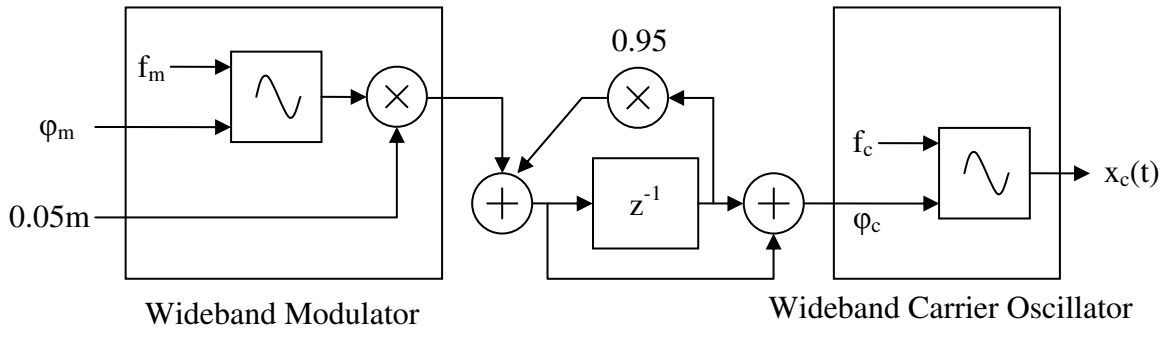


Fig. 56: Wideband Oscillator PM Setup

Phases often can be matched only for a single spectral component which results in a biased output. The bias is usually lowest when the fundamentals are in phase, implying a 90 degree shift between the oscillators for an integrated modulator signal. Since VA oscillators are expected to be biased anyway due to other processes like hard synchronization, it is good practice to insert a DC trap between the carrier oscillator output and subsequent stages.

Even if an oscillator lacks immediate phase control, it's still possible to add FM capabilities. As phase is the time integral of frequency, we may differentiate the modulator signal and add it to the frequency input (Fig. 57).

We should be aware that the discrete time difference is an approximation to the continuous time derivative. Similarly, the cumulative sum of the frequency input, which represents the phase in an accumulator-based oscillator, is not equal to the continuous time integral. Fortunately, these deviations exactly cancel.

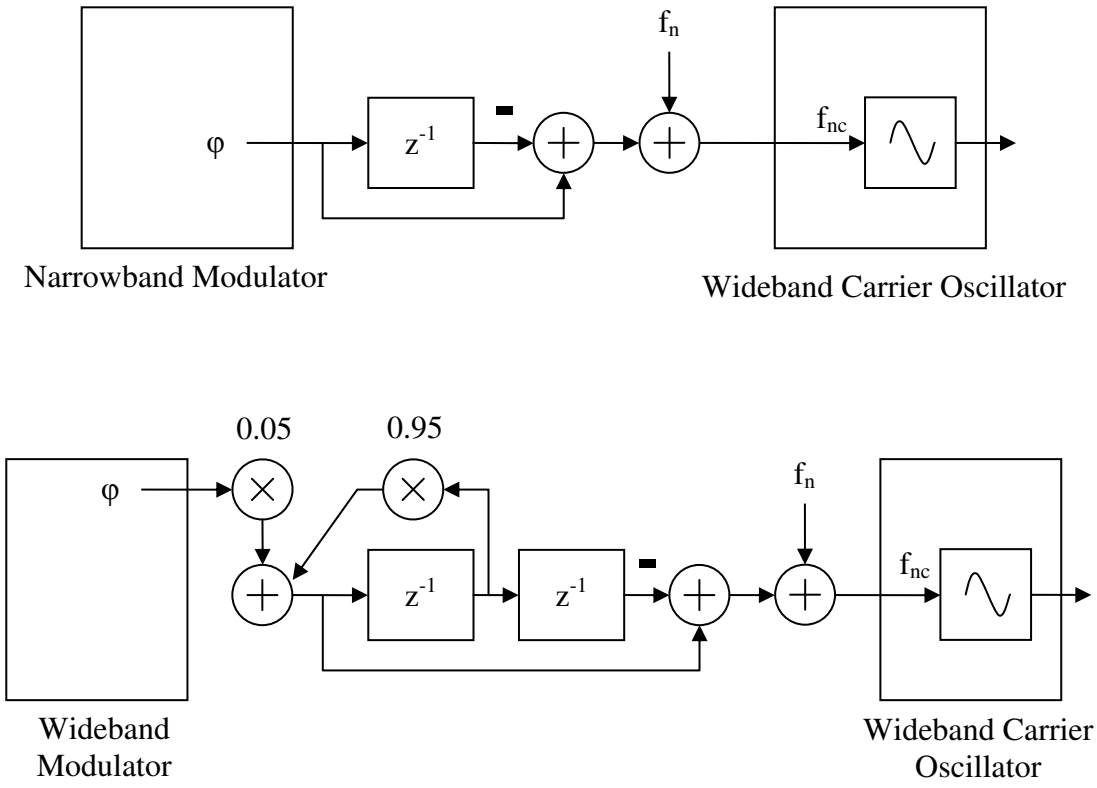


Fig. 57: FM Extension for Oscillators without Phase Input

1.10 Wavetable Oscillator

The wavetable oscillator is a standard block to generate arbitrary static harmonic spectra. Dynamic ones can be realized by crossfading the output of two units. It works like a sample-based oscillator that repeats a fixed length segment ad infinitum. Only low computational resources are required because we may design it to sound good with linear interpolation.

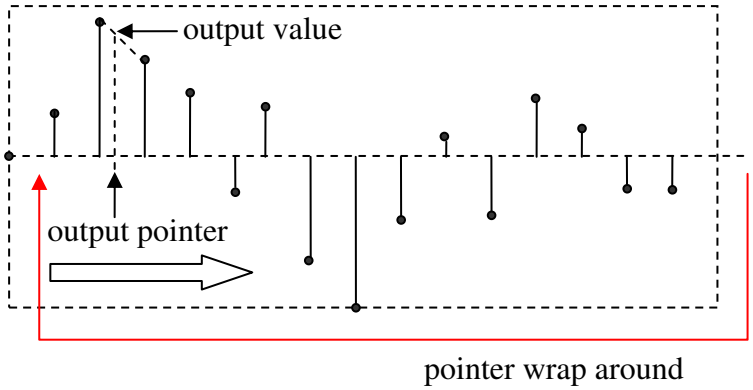


Fig. 58: Wavetable Oscillator Working Principle (N = 16)

The tabulated values x[n] and the output spectrum X[k] are directly related by the Discrete Fourier Transform (DFT):

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi j k n / N} \qquad x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{2\pi j k n / N}$$

When constructing a wavetable from the spectrum, a real valued signal must be ensured by satisfying the condition $X[k] = X^*[N-k]$ for every k. We can deliberately set the spectral components from k = 0 to N/2, whereby X[0] and X[N/2] should always be zero. The fundamental f_o of the output signal is given by the sample rate f_s, the table size N, and the fraction α of a table increment at which the output pointer proceeds. X[k] maps to the non-aliased part of the output spectrum S(f) as follows:

$S(\alpha f_s k/N) = X[k]$ for $0 \le k < N/2$ and k integer, 0 otherwise.

The special case α = 1 leads to the natural fundamental $f_{o(nat)} = f_s/N$ and harmonics at $kf_{o(nat)}$. Thus, α can also be interpreted as a transposition factor relative to $f_{o(nat)}$. If f_o falls below $f_{o(nat)}$, the sound starts missing highs, that's why the table should not be too short; a popular choice is N = 512.

Sometimes, the above formulae have to be evaluated quickly in large quantities. In this case, the Fast Fourier Transform (FFT) comes in handy and there are tricks that explore symmetry to speed up things further. These involve packing two real signals into a complex one, or two spectra into lower and upper halves. The reader may refer to DSP books for details.

Example sources for wavetables are:

- Editors
- Automatic extraction from samples. A purely time domain based approach is as possible [3] as spectral analysis and reordering [4].
- States of a slow system that is manipulated in real-time (→ Scanned Synthesis).
- Neural networks, cellular automata, whatever comes to our mind.

As long as the source delivers a natural spectrum with most power concentrated at lower harmonics, aliasing is normally not an issue. One word on the phase relations of harmonics:

Although often disregarded, they are important. Wavetables acquire a metallic-diffuse coloration which is audible up to a fundamental of several 100 Hz if the phases are spread randomly. We mentioned that linear interpolation is adequate to read out the table. How is that possible with a technique that performs rather poorly in sample-based oscillators? Let's start by showing the equivalence of linear interpolation to the convolution of the tabulated signal with a triangular window followed by resampling at the read location (Fig. 59).

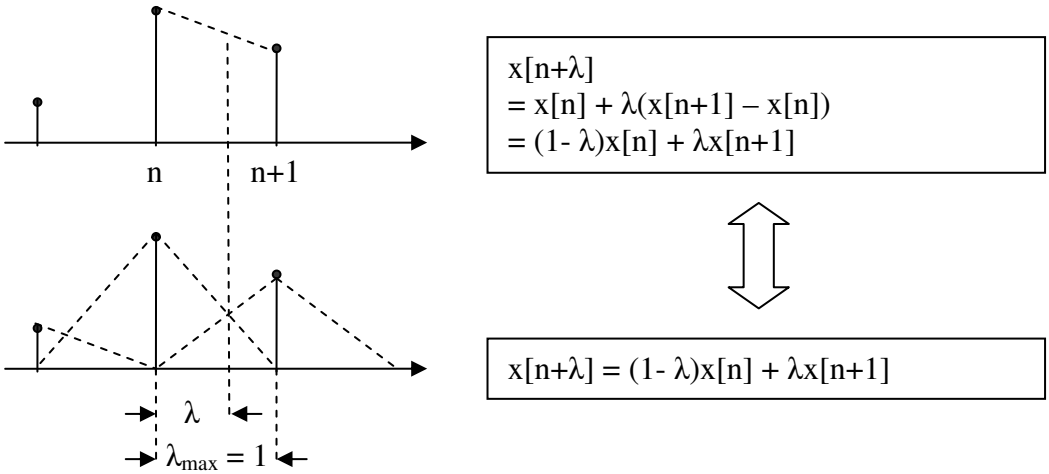


Fig. 59: Equivalence of Linear Interpolation and Convolution with a Triangular Window

This situation is exactly the same as with the sample-based oscillator except that the filter function is a triangular window instead of a bandlimited impulse (Fig. 60). It is of great advantage that we don't need to tabulate this window since the interpolated value can be calculated quickly without significant error. On the other side, the filter is far from perfect at higher frequencies. The amount of aliasing is derived with help of Fig. 60 and the fact that convolution with a triangular window in the time domain corresponds to multiplying the spectrum with the Fourier Transform S_w of the triangular window: $S_w(v) = \left[\frac{\sin(\pi v)}{\pi v}\right]^2$.

Any spectral component k within the range [0, N/2] of an N-size wavetable produces two first order aliased components with relative amplitudes given by:

$\gamma = \frac{|S_w((N \pm k)/N)|}{|S_w(k/N)|}$, approximated for k << N by $\gamma \approx \left[\frac{k}{N}\right]^2$.

In practical applications, the number of harmonics rarely exceeds $k_{max} = 250$, which paves a way to pin down aliasing below any desired level by increasing the table size to $N >> 2k_{max}$. This is just oversampling the tabulated signal.

We know from the section on sample-based oscillators that the transposition factor should be limited to $\alpha \approx [0.75, 2]$ for a system with $f_s = 48$ kHz and a spectrum up to 20 kHz to avoid a dull sound or aliasing. As a wavetable oscillator obeys the same rules, it would only be able to generate fundamentals in the vicinity of $f_{o(nat)}$ covering a range of about an octave. To alleviate that, different tables are selected depending on the desired fundamental.

In a first attempt, we could create a new table by halving its size and keeping the lower half of the spectrum. It will then have twice the natural fundamental and cover the octave above. Unfortunately, aliasing increases because typical spectra have the energy concentrated at low k. Additionally, a higher $f_{o(nat)}$ reduces auditory masking of aliased components below the fundamental and last but not least, the ear becomes more sensitive with increasing frequency.

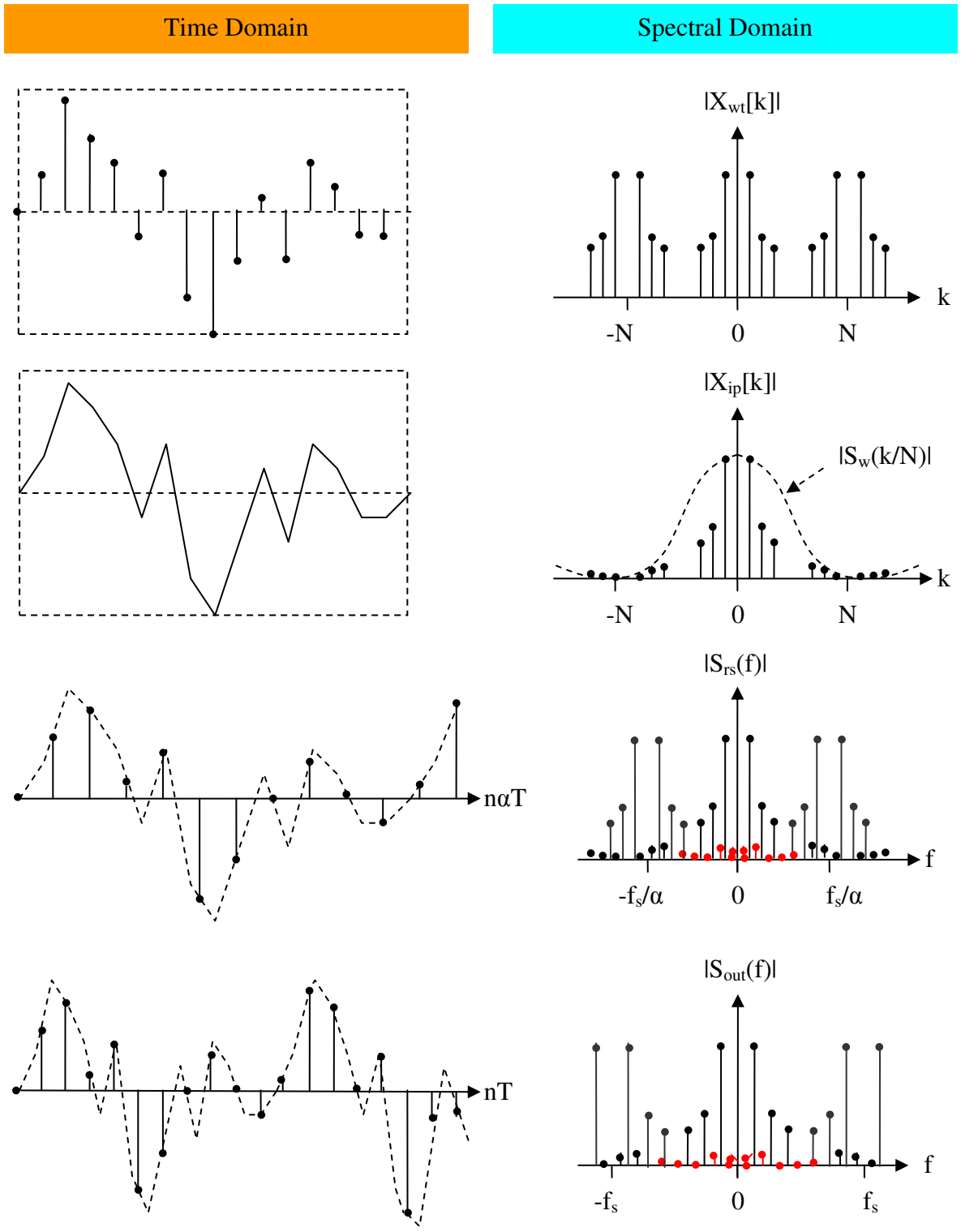


Fig. 60: Signals in the Wavetable Oscillator

In an attempt to improve, we keep the table size constant and zero the upper half of the preceding half-spectrum. This allows for doubling the transposition factor. It turns out that this oversampling scheme together with linear interpolation fits human perception very well. A wavetable oscillator that employs this principle is depicted in Fig. 61. To split the phase pointer into the integer table index and the fractional offset for interpolation (λ in Fig. 59), the same scheme as with the sample-based oscillator can be applied if the table size N is a power of two (see code at the end of section 1.4).

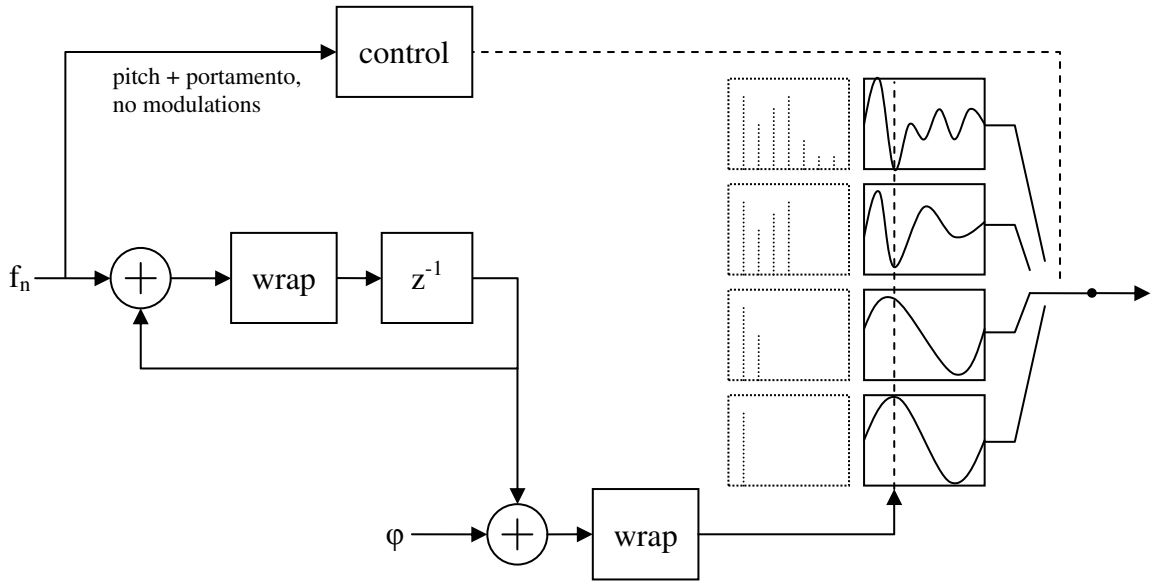


Fig. 61: Wavetable Oscillator

Low rate pitch modulation should not affect table selection to avoid periodic switching noise when the fundamental happens to be near a boundary. As this does not apply to the phase input, full FM capability at audio rates is preserved. In case of portamento, switching is unavoidable but not perceived normally. A single table may also cover less than an octave and the range of α needs not to be centred around $f_{o(nat)}$.

Design Example: $N = 512$, $f_s = 48$ kHz, $f_{audio} = 20$ kHz, $\alpha_{min} = 0.9$, $\alpha_{max} = 1.3$

→ $f_{o(nat)} = f_s/N = 93.75$ Hz, $\rho = \alpha_{max}/\alpha_{min} = 1.444$, $X[k] = 0$ for $N-k_{max} > k > k_{max}$

Table Index i	k_{max}	Frequency Range
0	213 (= $k_{max(0)} = Nf_{audio}/f_s$)	20 to 122 (= $r_o = \alpha_{max}f_{o(nat)}$)
1	147 (= $k_{max(0)}/\rho$)	122 to 176 (= ρr_o)
2	102 (= $k_{max(0)}/\rho^2$)	176 to 254 (= $\rho^2 r_o$)
3	71	254 to 367
4	49	367 to 531
5	34	531 to 766
6	23	766 to 1107
7	16	1107 to 1599
8	11	1599 to 2309
9	8	2309 to 3336
10	5	3336 to 4818
11	4	4818 to 6960
12	2	6960 to 10053
13	1	10053 to 20000

In synthesizers, the fundamental frequency f_o is obtained by exponentiation of the actual control variable. If this variable is used directly, no case differentiations are required as the table index can be calculated directly: $i = \max\left(0, floor\left[\frac{\log f_o - \log \alpha_{\min} - \log f_{o(nat)}}{\log \rho}\right]\right)$.

We finish this section with examples of aliasing analysis for a tabulated sawtooth.

Case 1: f_o = 500 Hz → k_{max} = 49

Signal power normalized to the fundamental: $P_s = \sum_{n=1}^{49}\left[\frac{1}{n}\right]^2 \approx \frac{\pi^2}{6}$

Aliased power normalized to the fundamental: $P_n \approx 2\sum_{n=1}^{49}\left[\frac{1}{n}\left(\frac{n}{N}\right)^2\right]^2 = \frac{80850}{N^4} \approx 1.18\cdot 10^{-6}$

SNR in dB = $10\log(P_s/P_n) \approx 58$ dB

No need to worry though; most aliasing occurs above the fundamental and is masked by the harmonics. If we assume a uniform noise power spectrum and take only components below half the fundamental frequency into account, we obtain a better estimate for the effectively perceived SNR:

$10\log(P_s/(P_n f_{crit}/(0.5 f_s))) = 10\log(P_s/(P_n\cdot 250/24000)) = 81$ dB.

The aliased spectrum is not continuous however, that's why we also have to examine the worst case, which occurs when the highest harmonic is aliased to a frequency below $f_o/2$:

$$P_{n(49)} = \frac{49^2}{N^4} \approx 3.49\cdot 10^{-8}$$

SNR = $10\log(P_s/P_{n(49)}) \approx 77$ dB

Case 2: f_o = 4000 Hz → k_{max} = 5

$$P_s = \sum_{n=1}^{5}\left[\frac{1}{n}\right]^2 \approx 1.46 \quad P_n \approx 2\sum_{n=1}^{5}\left[\frac{1}{n}\left(\frac{n}{N}\right)^2\right]^2 = \frac{110}{N^4} \approx 1.6\cdot 10^{-9}$$

SNR ≈ 89 dB

Conclusion:

The wavetable oscillator in Fig. 61 shows better spectral performance at higher fundamentals. This unusual behaviour is attributed to the combination of oversampling proportional to the fundamental frequency and constant output bandwidth. As long as the power is concentrated at low harmonics, everything is fine, but noise may become an issue with tabulated band pass spectra. In this case, enlarging the table will solve the problem at the expense of increased memory requirements. The popular choice of N = 512 is scarcely sufficient, however, for state-of-the-art quality, especially when replacing VA oscillators, N = 1024 is recommended.

Some remarks on the output spectrum:

Linear interpolation has a Sinc^2 frequency response. When tables are constructed from spectra, we have to emphasize the k-th spectral component by

$S_{pre}[k] = \left[\frac{\pi k/N}{\sin(\pi k/N)}\right]^2$ in order to preserve the highs.

The same applies to a signal that is extracted in the time domain. Prefiltering it with

$H_{pre}(z) = \frac{1}{0.75 + 0.25 z^{-1}}$ before the table segments are gathered is a practical solution.

1.11 Hard Synchronization

Hard synchronization (a.k.a. Sync) of two oscillators descends from classic analog synthesizers. Whenever the master oscillator completes a cycle, the slave oscillator is reset to the beginning of its waveform (Fig. 62). While the sound of synced pulses is characteristic, not to say a cliche, the combination of Sync and FM is one of the most fruitful ad-hoc synthesis techniques. In the time domain, a synced signal can be described by multiplying the slave oscillator signal with a shifted rectangular window and repeating the resulting segment at the master's rate (Fig. 63). In the frequency domain, this corresponds to the convolution of the slave oscillator spectrum with the shifted window spectrum (whose magnitude is a Sinc function in this case) followed by resampling the continuous spectrum at the master oscillator frequency.

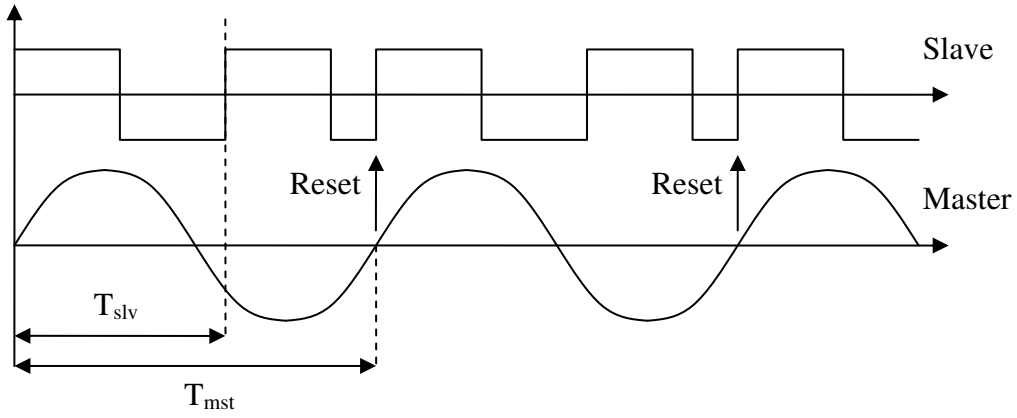


Fig. 62: Hard Sync Oscillator Signals

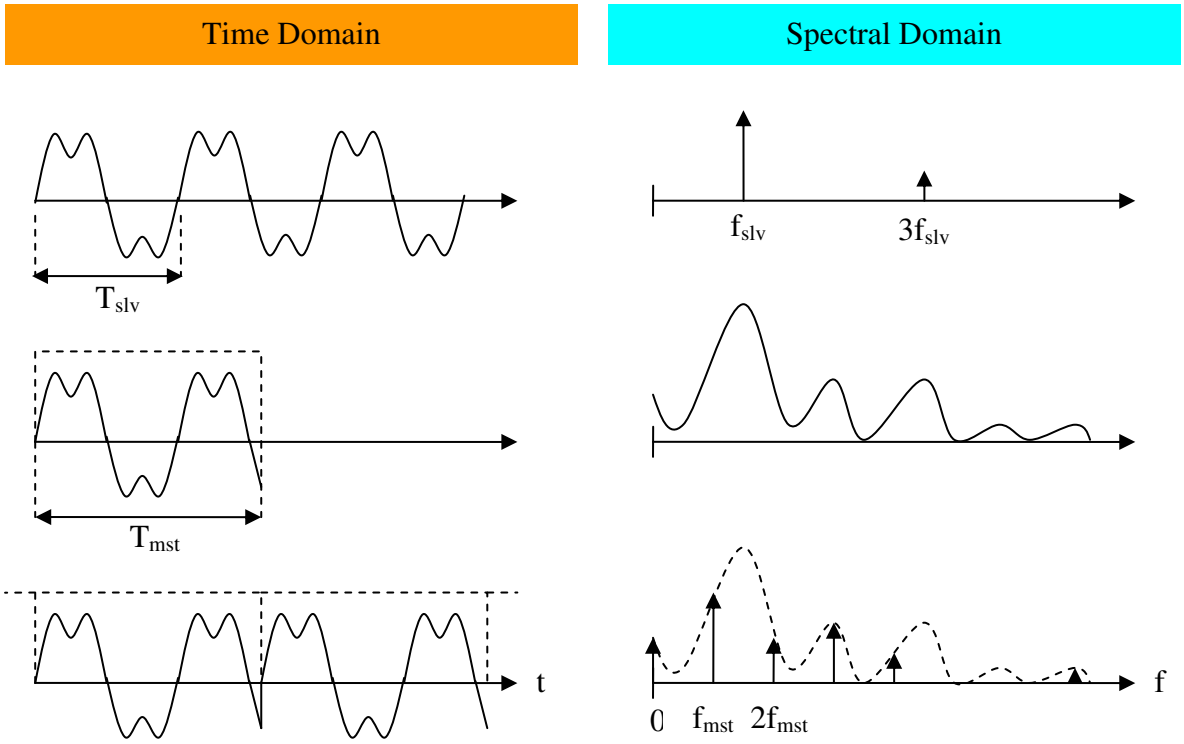


Fig. 63: Hard Sync Signals and Spectra

Assuming an unsynchronized slave oscillator signal s(t) with the complex spectrum $S_{orig}(f) = \sum_{k=-\infty}^{\infty} c_k \delta(f - kf_{slv})$, the output spectrum of the synchronized signal becomes:

$$S_{sync}(f) = \sum_{n=-\infty}^{\infty} \delta(f - nf_{mst}) \sum_{k=-\infty}^{\infty} c_k \frac{\sin(\pi T_{mst}(f - kf_{slv}))}{\pi T_{mst}(f - kf_{slv})} e^{-j\pi T_{mst}(f - kf_{slv})}$$

Discussion:

1. The synced signal spectrum has only components at integer multiples of the master oscillator frequency and f = 0.
2. Their magnitude is determined by a sum of phase shifted components derived from the unsynchronized signal. These components are copies of the rectangular window spectrum centred at the frequency of spectral peaks in the unsynchronized signal and weighted by their magnitude.
3. The bandwidth of the output signal is the bandwidth of the original signal plus the bandwidth of the window spectrum.

As the Sinc function has infinite support and diminishes slowly, aliasing will occur in a discrete time realization. We solve this problem by changing the window from a rectangle to a function with steeper roll-off and low side lobes. On a 96 kHz system, the rectangle can be convolved with a bandlimited impulse, which results in a function that preserves the original characteristics of Sync as much as possible without being susceptible to aliasing (Fig. 64). This is equivalent to replacing the edges by a BLEP [5].

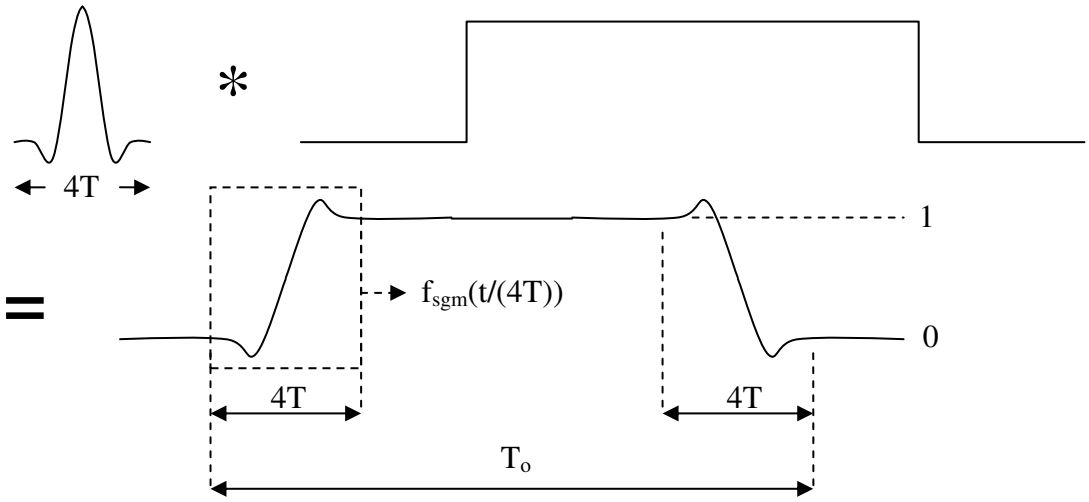


Fig. 64: Bandlimited Rectangle Window for Hard Sync on 96 kHz Systems

Besides band limiting, there's a second challenge: We have to detect the zero crossings of the master oscillator and since they usually do not coincide with a sample point, we also need to set the new phase of the slave depending on the master's phase to $\varphi_{slv} = \varphi_{mst} \cdot f_{slv}/f_{mst}$.

That's the reason why immediate phase control is vital to slave oscillators. A realization of a synced oscillator pair is shown in Fig. 65. We may feed the waveform output of the master into the phase input of the slave using a wideband FM circuit to get the exciting combination of FM and Sync. In the slave oscillator, it's important to place any building blocks with memory or feedback after the multiplication with the windowed signal if we don't reset their states. Sync produces bias: A DC trap should be added to the output signal chain.

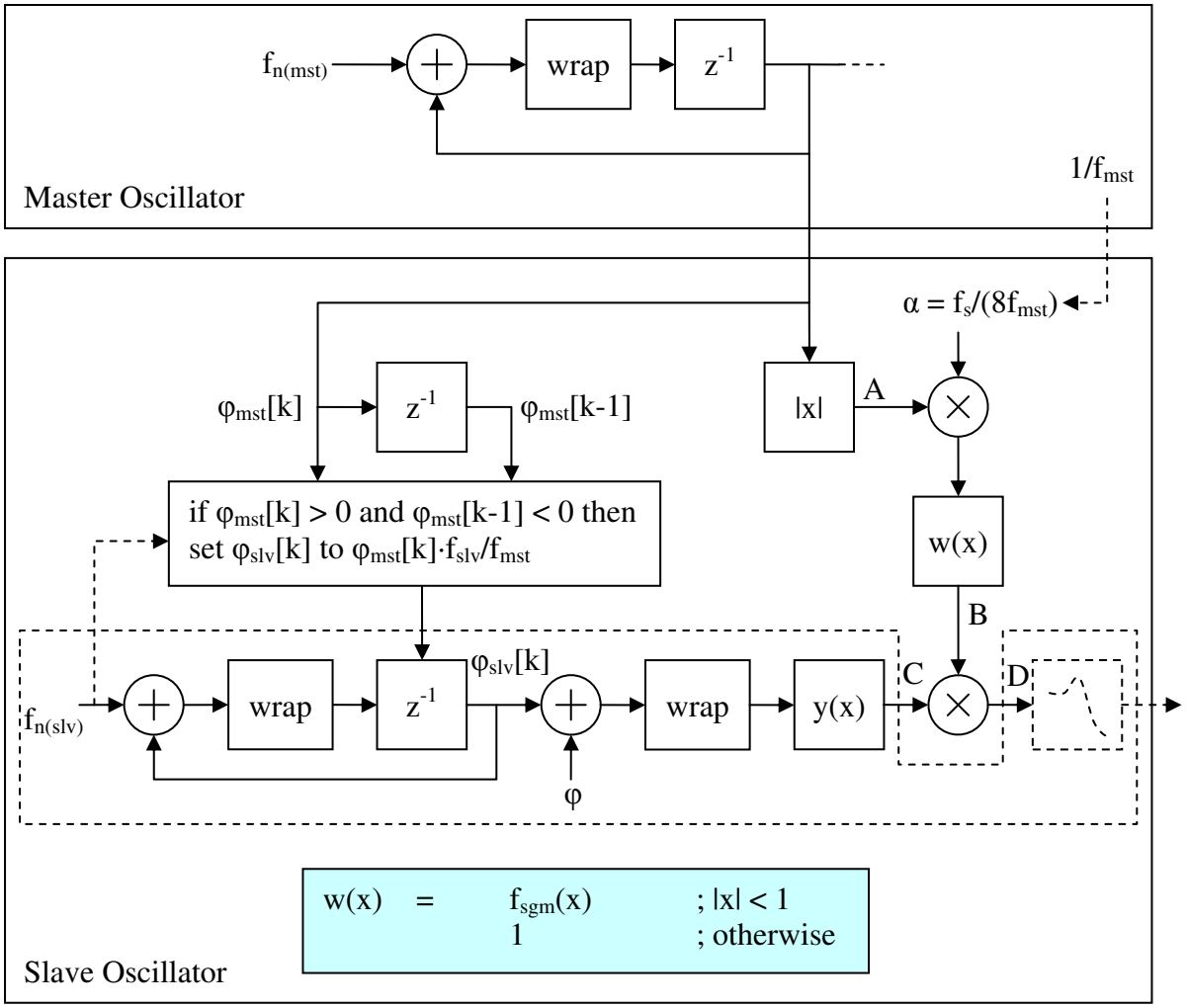


Fig. 65: Hard Sync Oscillator Pair

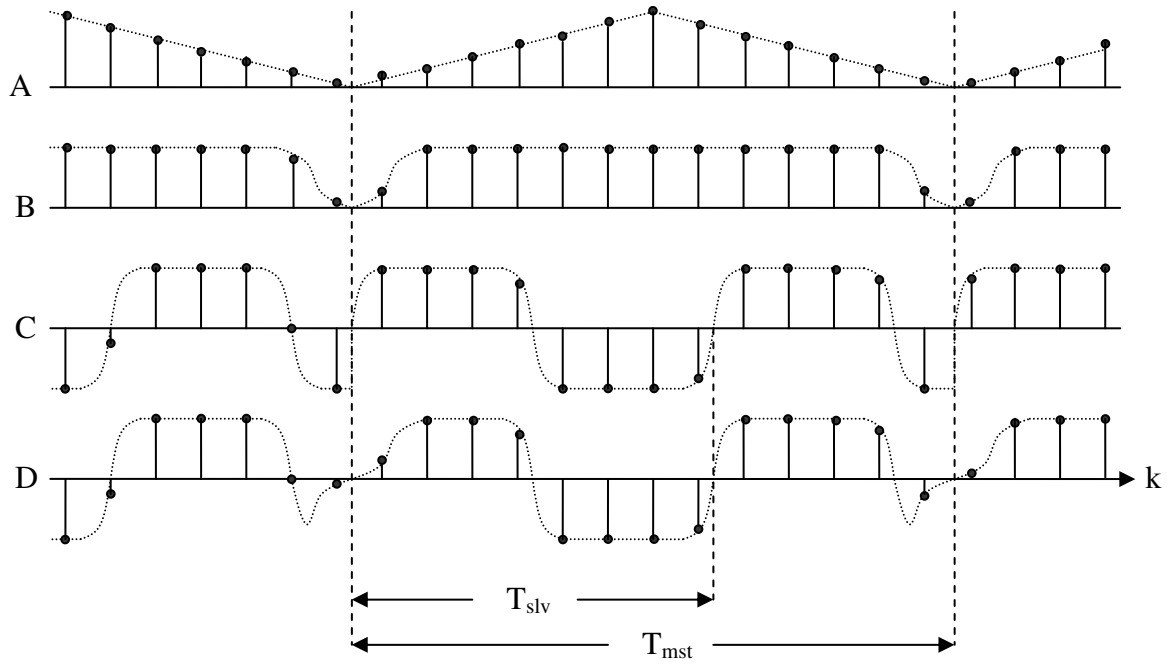


Fig. 66: Signals in the Synced Slave Oscillator

At point A, a triangle in the interval [0, 1] at the master's fundamental frequency is present. If we want to tweak the system to make it work satisfactorily at f_s = 48 kHz, the factor α is crucial as it determines the slope at which the window function is evaluated.

Typical modifications are:

- Scaling: A lower value reduces the slope and proportionally the bandwidth of the window. This results in less aliasing, but the sound becomes peaky for large slave to master frequency ratios.
- Setting a lower limit of 1 to ensure that the window function completes a whole cycle even at the maximum master frequency.

Another field of experimentation is the window w(x) with focus on polynomial segments whose spectrum rolls off quickly when used as an edge replacement. The simplest function is a linear ramp, which results in a triangle window. It can be shown (by iterated differentiation and remembering that the Dirac delta has a flat spectrum) that if the n-th lowest order derivatives are zero at the borders, the spectrum asymptotically declines at $1/f^{n+2}$. The lowest order polynomial with zero first derivatives that satisfies all side conditions is $3x^2 - 2x^3$.

The following design example is suggested for f_s = 48 kHz:

$$\alpha = \min\left(8, \max\left(1, \frac{f_s}{24 f_{mst}}\right)\right) \qquad w(x) = \begin{cases} 3x^2 - 2x^3 & ;x < 1 \\ 1 & ;otherwise \end{cases}$$

All spectra are taken with a sinusoidal slave oscillator. For a more complex signal, its spectral components could be analyzed separately and summed up. When we consider the spectral envelope of typical oscillator signals, an aliased component below $f_{mst}/2$ can be expected to be down by around 80 dB relative to the desired signal. While this situation is satisfactory, it worsens significantly for master frequencies at the top of the audio range. In this case, it's preferable to fold down f_{mst} by an octave when it would exceed 6 kHz.

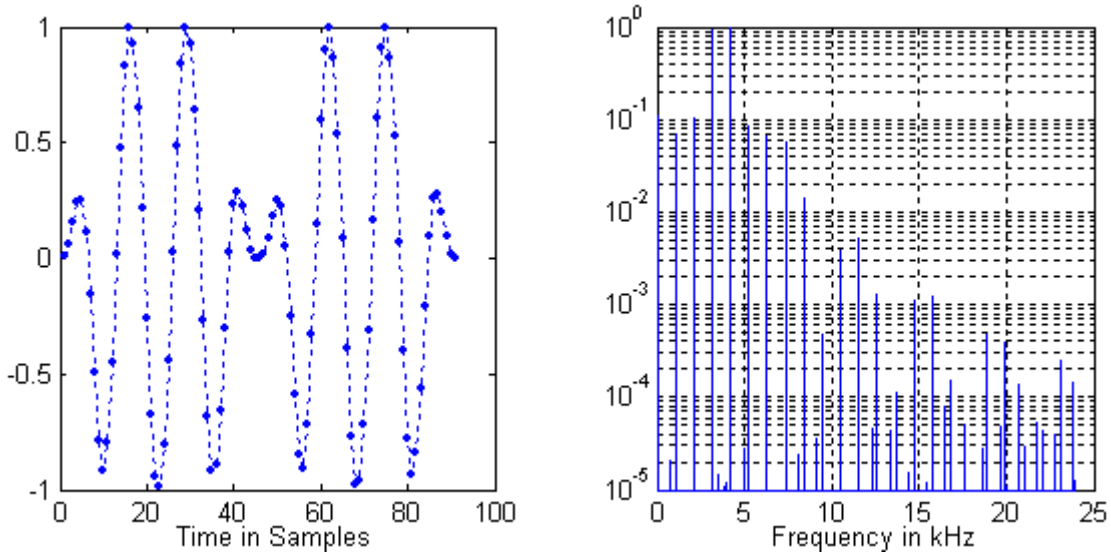


Fig. 67: Hard Sync Oscillator Spectrum, f_{mst} = 1050 Hz, f_{slv} = 3700 Hz

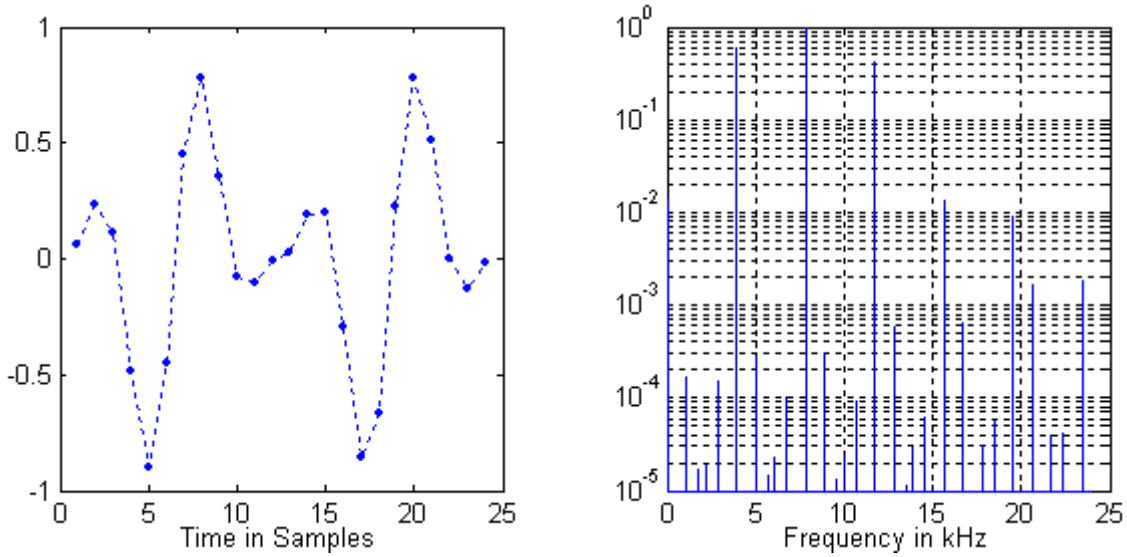


Fig. 68: Hard Sync Oscillator Spectrum, f_{mst} = 3910 Hz, f_{slv} = 7400 Hz

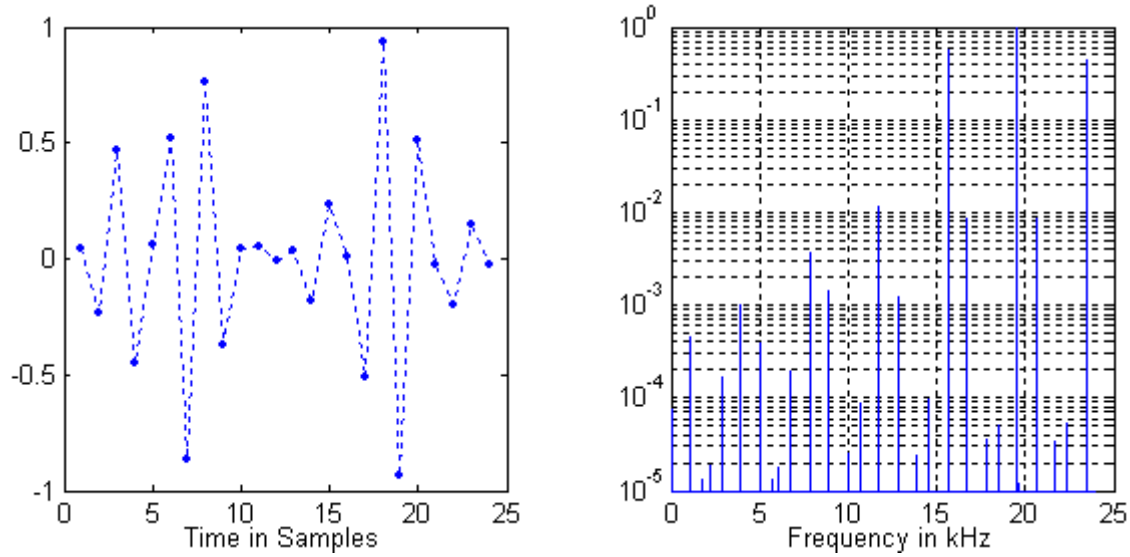


Fig. 69: Hard Sync Oscillator Spectrum, f_{mst} = 3910 Hz, f_{slv} = 19300 Hz

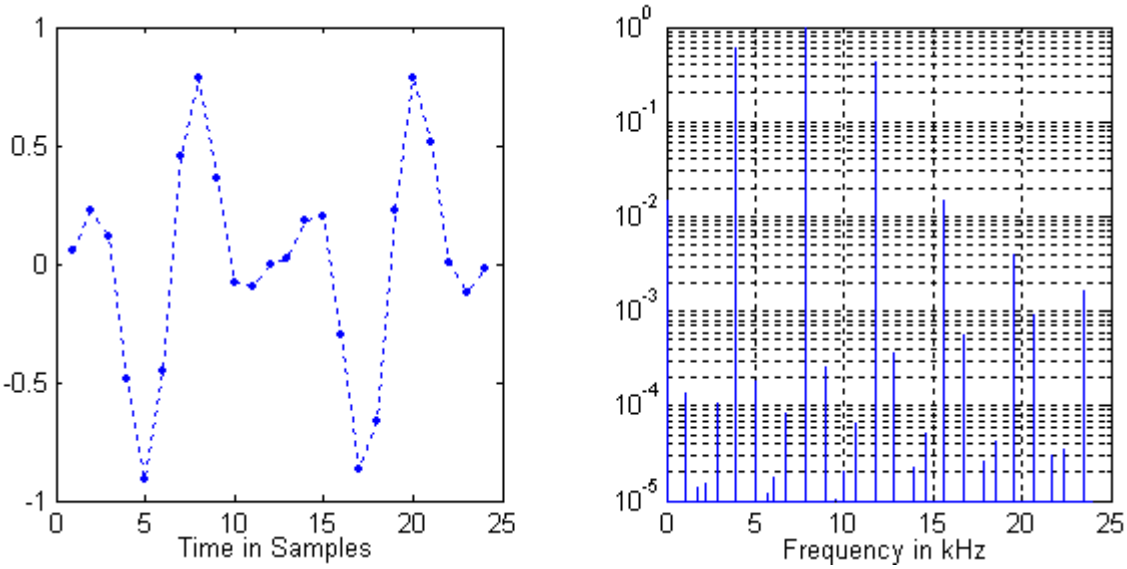


Fig. 70: Hard Sync Oscillator Spectrum, w(x) =Raised Cosine Window, f_{mst} = 3910 Hz, f_{slv} = 7400 Hz

Some existing designs employ a raised cosine window that also has zero first and non-zero second derivatives at the borders:

$$w(x) = \begin{cases} \frac{1-\cos(\pi x)}{2} & ; x < 1 \\ 1 & ; otherwise \end{cases}$$

While being computationally more expensive, its spectral characteristics surpass the polynomial solution only marginally (Fig. 70).

Summary:

Although Sync is feasible in a 48 kHz system, this is one of the cases where a double rate system has audible advantages. There's a trade off between aliasing and sonic authenticity for high slave to master frequency ratios. The analog original retains a pleasant broadband formant characteristic while the digital version starts to sound like a single resonant peak due to the narrow bandwidth of the window (which is essential to limit aliasing).

2 Alternative and Specialized Oscillators

2.1 Sinusoidal Oscillators based on Second Order Systems

This class of sinusoidal oscillators is the most efficient for fixed parameters. On the other hand, phase control involves the calculation of trigonometric functions to high precision rendering audio rate modulation laborious. Even maintaining a smooth signal and constant amplitude when the frequency is changed is not trivial.

2.1.1 Coupled Form Oscillator

An archetype of the aforementioned oscillator class is obtained by encoding the elements of a rotating vector into the state variables of a second order system (Fig. 71). The resulting oscillator has two outputs whose phases differ by 90°, hence sine and cosine functions are generated simultaneously. The **phase increment φ** is proportional to the frequency: $\boldsymbol{\varphi = 2\pi f_o/f_s}$. This relation is used throughout the entire section.

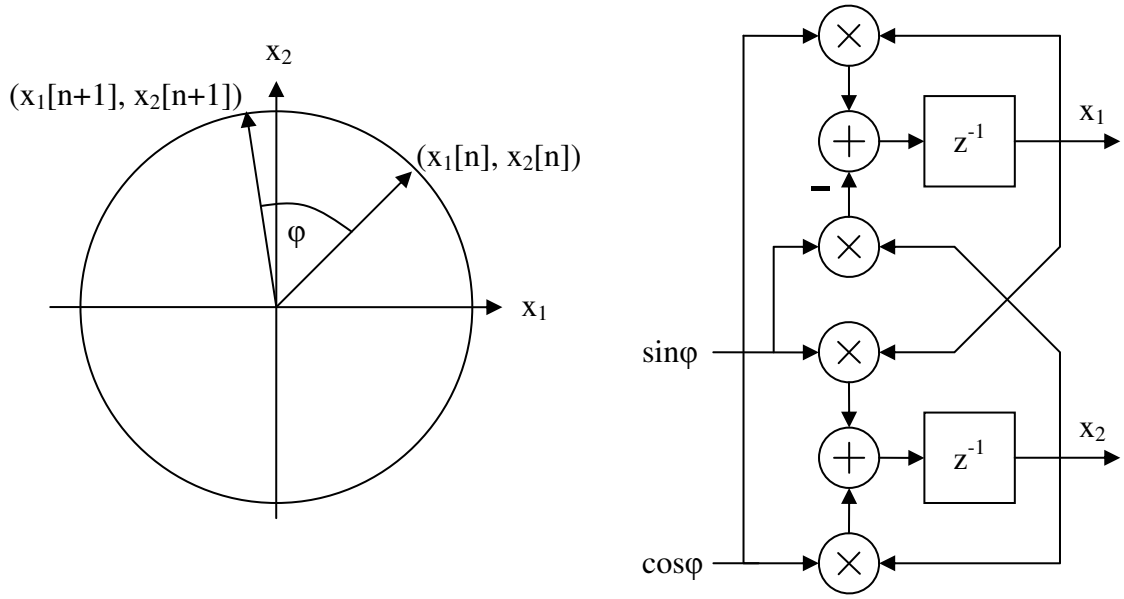


Fig. 71: Coupled Form Sinusoidal Oscillator

$$\mathbf{x}[n+1] = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \mathbf{x}[n]$$; State Space Equation of the Coupled Form Oscillator

Suggested initial conditions: $\mathbf{x} = (1, 0)$.

The structure is insensitive to both coefficient and state quantization. It exhibits neither glitches nor permanent changes of the amplitude when φ is modulated. In a practical realization, the amplitude drifts with time due to roundoff errors. As a workaround, we may scale the coefficients by a factor slightly above 1 and use saturation arithmetic. Another method is to approximately normalize the state vector from time to time as follows:

$$\alpha = (3 - |\mathbf{x}|^2)/2 \qquad \alpha\mathbf{x} \rightarrow \mathbf{x}$$

2.1.2 Direct Form Oscillator

Another oscillator type is found by plugging x[n] = sin(nφ + φ$_o$) into the trigonometric identity sin(a+b) + sin(a-b) = 2sinacosb. We get: x[n+1] + x[n-1] = 2x[n]cosφ.
A direct form one topology is used to implement the equation (Fig. 72).

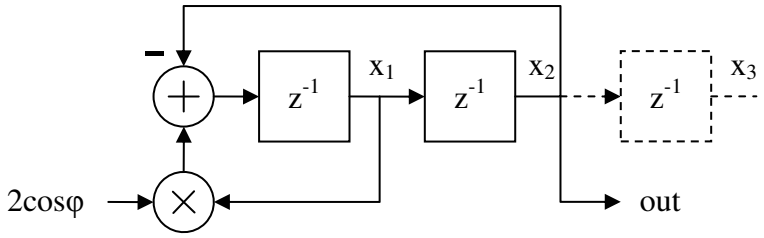


Fig. 72: Direct Form Sinusoidal Oscillator

$$\mathbf{x}[n+1] = \begin{bmatrix} 2\cos\varphi & -1 \\ 1 & 0 \end{bmatrix} \mathbf{x}[n]$$; State Space Equation of the Direct Form Oscillator

Suggested initial conditions, must be set whenever φ changes: **x** = (1, cos φ) or (sin φ, 0).

This is the efficiency champion among sinusoidal oscillators. However, its application range is limited because the amplitude exhibits a discontinuity and undergoes a permanent change when φ is set to a new value without an accompanying state update. Furthermore, the use of single precision arithmetic for φ < 0.003 causes the SNR to drop below 80 dB and introduces an audible tuning error due to coefficient quantization. In order to solve the amplitude issue, we first note that the desired new value of x_1 is linked with x_2 by the phase increment φ_{new}:

Combining $x_1 = \sin(n\varphi + \varphi_0 + \varphi_{new})$ and $x_2 = \sin(n\varphi + \varphi_0)$ yields
$x_1 = \sin(n\varphi + \varphi_0)\cos\varphi_{new} + \cos(n\varphi + \varphi_0)\sin\varphi_{new}$ and finally
$x_1 = x_2 \cos\varphi_{new} \pm \sqrt{1 - x_2^{\,2}} \sin\varphi_{new}$.

The new output amplitude is 1; consequently this should also have been the case before (as with the suggested initial conditions). Roundoff errors may lead to excess amplitude, so we have to limit $|x_2|$ to 1 before taking the root. Regular state updates without actually changing the frequency help to stabilize the amplitude. Determining the sign is not trivial: Probably the best way is to add another delay element. Before the frequency update, the states are:

$x_1 = \sin((n+1)\varphi + \varphi_0)$ and $x_3 = \sin((n-1)\varphi + \varphi_0)$. Therefore
$x_1 - x_3 = 2\cos(n\varphi + \varphi_0)\sin\varphi$, which has the sign of cos(nφ+φ$_o$).

Whenever φ changes, we just set the new state x_1 according to the formula
$x_{1(new)} = x_2 \cos\varphi_{new} + \mathrm{sgn}(x_1 - x_3)\sqrt{1 - x_2^{\,2}} \sin\varphi_{new}$ with $\varphi_{new} = 2\pi f_{o(new)}/f_s$
before the next value of x_1 is calculated. The oscillator will then seamlessly proceed at the new frequency. There must be at least one unit delay between consecutive changes lest the value of x_3 is incorrect. Rewriting the equation, only the cosine function has to be computed:

$$x_{1(new)} = x_2 \cos\varphi_{new} + \mathrm{sgn}(x_1 - x_3)\sqrt{(1 - x_2^{\,2})(1 - \cos^2\varphi_{new})}.$$

2.1.3 Chamberlin Oscillator

Another interesting approach is the state variable filter described in [6] with Q set to infinity (Fig. 73).

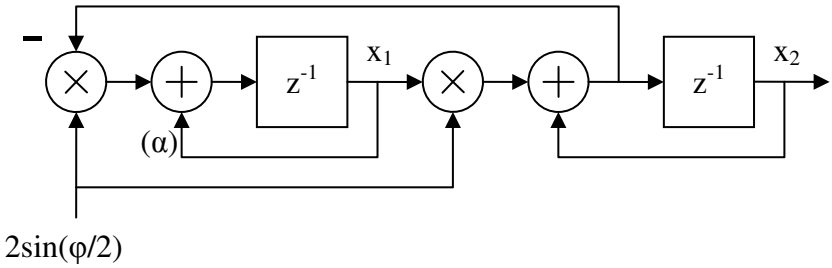


Fig. 73: Chamberlin Sinusoidal Oscillator

$$\mathbf{x}[n+1] = \begin{bmatrix} 1-\left(2\sin\frac{\varphi}{2}\right)^2 & -2\sin\frac{\varphi}{2} \\ 2\sin\frac{\varphi}{2} & 1 \end{bmatrix} \mathbf{x}[n] \quad \text{; SSE of the Chamberlin Oscillator}$$

Suggested initial conditions: $\mathbf{x} = (1, -\sin(\varphi/2))$ or $(0, \cos(\varphi/2))$.

Both eigenvalues $\lambda_{1,2} = e^{\pm i\varphi}$ are complex conjugate and located on the unit circle, hence the system actually oscillates at $f_o = \varphi f_s/(2\pi)$. At low frequencies, the outputs are approximately orthogonal. The exact phase relationship is obtained from the transfer function:

$$H_{12}(z) = \frac{2\sin(\varphi/2)z^{-1}}{1-z^{-1}} \qquad \phi_{12}(\varphi) = \arg\left(H_{12}(e^{i\varphi})\right)$$

$$\phi_{12}(\varphi) = \arg\left[\frac{2\sin(\varphi/2)e^{-i\varphi}}{1-e^{-i\varphi}}\right] = \arg\left[\frac{2\sin(\varphi/2)e^{-i\varphi/2}}{e^{i\varphi/2}-e^{-i\varphi/2}}\right] = \arg\left[\frac{2\sin(\varphi/2)e^{-i\varphi/2}}{2i\sin(\varphi/2)}\right] = -\frac{\pi+\varphi}{2}$$

We also see that both outputs have identical amplitudes: $\left|H_{12}(e^{i\varphi})\right| = 1$.

If we want perfect orthogonality, the pair $x_1[n]$ and $x_{2m}[n] = (x_2[n] + x_2[n+1])/2$ does the job at the expense of a slight amplitude difference: $\left|H_{12m}(e^{i\varphi})\right| = \cos(\varphi/2)$.

Like the Coupled Form type, this oscillator is insensitive to quantization errors and works well with single precision arithmetic down to the infrasonic range. Moreover, only one tuning coefficient is required. Frequency changes do not induce a discontinuity at the outputs as they are tapped off integrators - however, a minor amplitude change occurs due to the frequency dependent deviation from orthogonality. Saturation arithmetic combined with slightly increased feedback by setting α to $1 + 0.01\varphi$ instead of 1 eliminates this effect and any amplitude drift caused by roundoff errors. If saturation is applied only to x_1, the resulting harmonics are reduced in x_2 by the low pass action of the integrator.

Depending on the desired tuning accuracy and the ratio of maximum frequency to sample rate, one may experiment substituting polynomials for trigonometric functions or even 1 for cosφ and φ for sinφ in the Coupled Form and the Chamberlin oscillator. In the former, the amplitude must be limited by clipping or saturation as the approximation moves the poles outside the unit circle. Immediate absolute phase control is possible by setting the states directly using trigonometric functions.

2.2 Oscillators based on Discrete Summation Formulae (DSF)

In section 1.5 we adopted the concept of a bandlimited impulse train (BLIT) to derive classic waveforms. There's a way to calculate an ideal BLIT by combining a summation formula for the Geometric series with Euler's formula [7]:

$$blits(t,a,N) = \sum_{n=1}^{N} a^n \sin(n\omega t) = \sum_{n=1}^{N} a^n \frac{\left[e^{nj\omega t} - e^{-nj\omega t}\right]}{2j} = \sum_{n=1}^{N} \frac{\left[e^{n(j\omega t+\ln a)} - e^{-n(j\omega t+\ln a)}\right]}{2j}$$

$$blits(t,a,N) = \frac{1}{2j}\sum_{n=1}^{N} e^{n(j\omega t+\ln a)} - \frac{1}{2j}\sum_{n=1}^{N} e^{n(-j\omega t+\ln a)} = \frac{1}{2j}\cdot\left[\frac{1-e^{[N+1](j\omega t+\ln a)}}{1-e^{j\omega t+\ln a}} - \frac{1-e^{[N+1](-j\omega t+\ln a)}}{1-e^{-j\omega t+\ln a}}\right]$$

$$blits(t,a,N) = \frac{a\sin(\omega t) - a^{N+1}\sin((N+1)\omega t) + a^{N+2}\sin(N\omega t)}{1-2a\cos(\omega t)+a^2}, \text{ and in an analogous way}$$

$$blitc(t,a,N) = \sum_{n=1}^{N} a^n \cos(n\omega t) = \frac{1-a\cos(\omega t) - a^{N+1}\cos((N+1)\omega t) + a^{N+2}\cos(N\omega t)}{1-2a\cos(\omega t)+a^2} - 1$$

If we set a = 1, the BLIT has a flat spectral envelope. Converting the difference of the highest frequency terms to a product and expressing the rest as half angle functions yields:

$$blitc(t,N) = \sum_{n=1}^{N}\cos(n\omega t) = \frac{1}{2}\left[\frac{\sin((N+0.5)\omega t)}{\sin(\omega t/2)} - 1\right]$$

$$mblits(t,N) = \frac{\sin(N\omega t)}{2} + \sum_{n=1}^{N-1}\sin(n\omega t) = \frac{1}{2}[1-\cos(N\omega t)]\cot(\omega t/2)$$

$$mblitc(t,N) = \frac{\cos(N\omega t)}{2} + \sum_{n=1}^{N-1}\cos(n\omega t) = \frac{1}{2}[\sin(N\omega t)\cot(\omega t/2) - 1]$$

The first expression can be evaluated with reasonable effort on most floating point processors using two linearly interpolated sine table lookups and a division. The number of harmonics N is set whenever the fundamental frequency changes but does not follow pitch modulation (the same scheme as with wavetable synthesis). The main advantage of such a DSF-BLIT is the complete absence of aliasing at any fundamental in the whole audio band. Unfortunately, there's no closed-form solution to produce time integrated descendants to emulate classic waveforms. Instead, they must be generated by subsequent integration sacrificing immediate phase control.

Remarkably, some expressions are free of the time consuming division. On the other side, the cotangent is singular at integer multiples of π and the function has maxima within an order of magnitude of N. For the following, we concentrate on mblitc(t,N), because this function results in a bandlimited sawtooth when integrated. A different factorization eliminates the singularity. After amplitude normalization and substituting φ for ωt, we get:

$$mblitcn(\varphi,N) = \frac{1}{N}\left[\frac{\cos(N\varphi)}{2} + \sum_{n=1}^{N-1}\cos(n\varphi)\right] = \left[\frac{\sin(N\varphi)}{N\varphi}\right]\left[\frac{\varphi}{2}\cot\left(\frac{\varphi}{2}\right)\right] - \frac{1}{2N}$$

This function is even and periodic in 2π. Therefore, φ can be confined to the range [-π/2, π/2] for evaluation. Both products are limited to a magnitude of 1. While the second term resembles an upside-down parabola and calls for polynomial approximation, the Sinc would require an impractically large table, even with linear interpolation, because N hits 1000 if the oscillator operates at a fundamental of 20 Hz. We already know that this kind of problem can

be solved by windowing at the expense of a broader spectrum. It turns out that a train of bandlimited impulses from section 1.3 is more efficient in terms of arithmetic operations and memory, but since the windowed DSF-BLIT approach still crops up now and then, we nevertheless give an example.

A limit of $|N\varphi| < 12\pi$ combined with a simple approximation for xcot(x) has proven sufficient to keep aliasing down by 80 dB up to a fundamental of 4 kHz on a system with $f_s = 48$ kHz. Low aliasing at higher fundamentals would require a more precise approximation. For compactness, we normalize and wrap φ to [-1,1] to get the windowed BLIT system listed below with the spectrum of Fig. 74.

$$w(N\varphi) = \begin{cases} \cos^2\left(\frac{\pi}{2}\left[\frac{N\varphi}{12}\right]^2\right) & \text{for } |N\varphi| < 12 \\ 0 & \text{otherwise} \end{cases}$$

$$s(N\varphi) = w(N\varphi)\frac{\sin(\pi N\varphi)}{\pi N\varphi}$$

$$mblitcn(\varphi, N) \approx s(N\varphi)\left[1 - \varphi^2\right] - \frac{1}{2N}$$

Phase Update :

$$\varphi[n+1] = \varphi[n] + \Delta\varphi$$

$$\text{if } \varphi[n+1] > 1 \text{ then } \varphi[n+1] = \varphi[n+1] - 2$$

Frequency Update :

$$\Delta\varphi = 2\pi f_o / f_s$$

$$N = floor(22000 / f_o)$$

When the function s(Nφ) is tabulated, at least 30000 entries for one of the symmetrical halves are recommended for non-interpolated readout; about 1000 are sufficient if linear interpolation is applied.

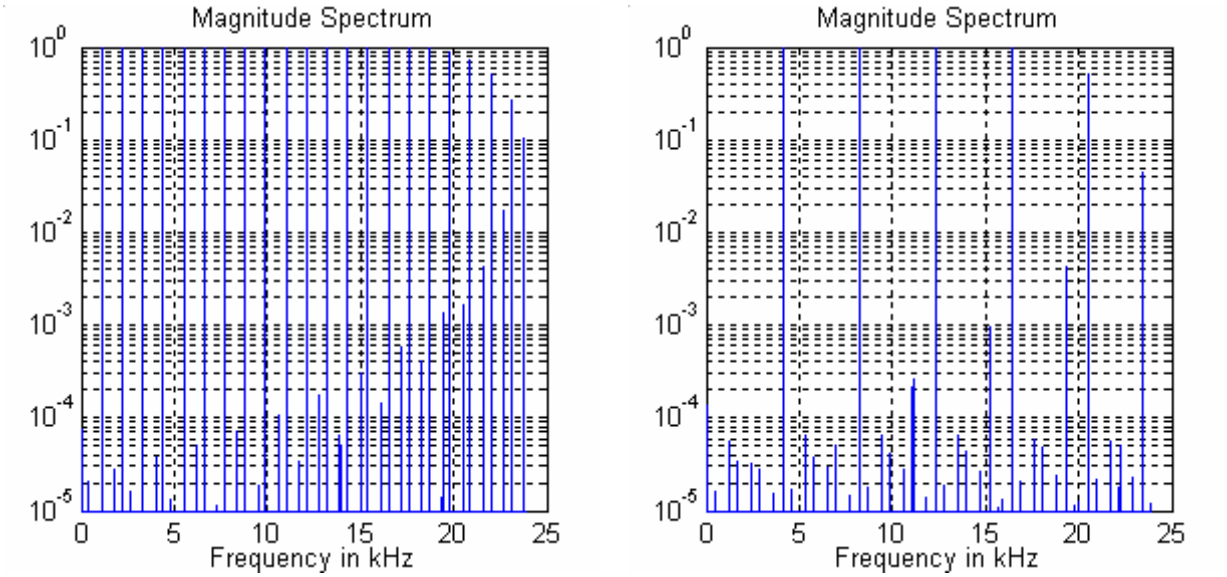


Fig 74: Windowed DSF-BLIT Spectra

A BLIT is typically used as an excitation signal for resonating systems like formant and comb filters or physical models. As we already know, it can also be integrated to make a sawtooth. Such a system is shown in Fig. 75 and employs a low pass filter as a leaky integrator, whose frequency tracking prevents amplification of residual bias and aliased components below the fundamental.

To conclude the foregoing, the DSF-BLIT oscillator is mainly attractive if division is executed fast and the BLIT is used in its original form. It is also first choice in applications where a perfectly pure or exponentially decaying spectrum is essential.

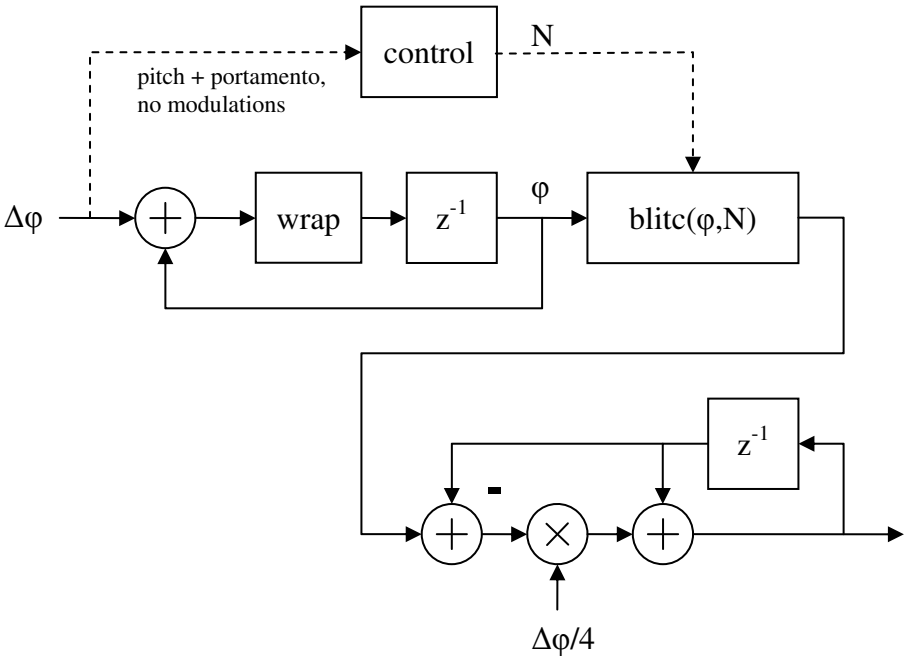


Fig. 75: DSF-BLIT Sawtooth Oscillator

2.3 Oscillators based on Integrated Prototype Signals

2.3.1 Differentiated Parabolic Wave (DPW) Oscillator

These oscillators have been introduced in [15] and attract some interest due to a complexity between the naive and the BLIT-based approach. A train of parabolic segments can be constructed in a way that the harmonics decrease at $1/f^3$ thus reducing the risk of fold-over. This signal is then differentiated to get the desired shape. In a discrete time realization, finite difference is used to approximate continuous time differentiation. Since both operations are linear, no aliasing occurs at this stage. Furthermore, their high pass characteristic helps to suppress aliased components below the fundamental. The triangle oscillator is an especially useful embodiment of this concept since the signal has low harmonic content and alternatives are computationally intensive.

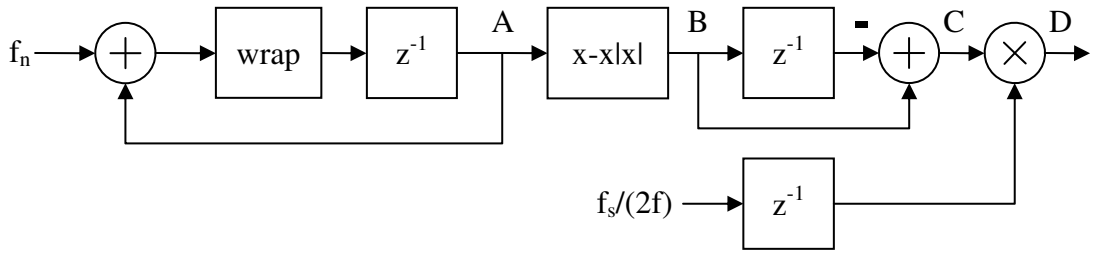


Fig. 76: DPW Triangle Oscillator

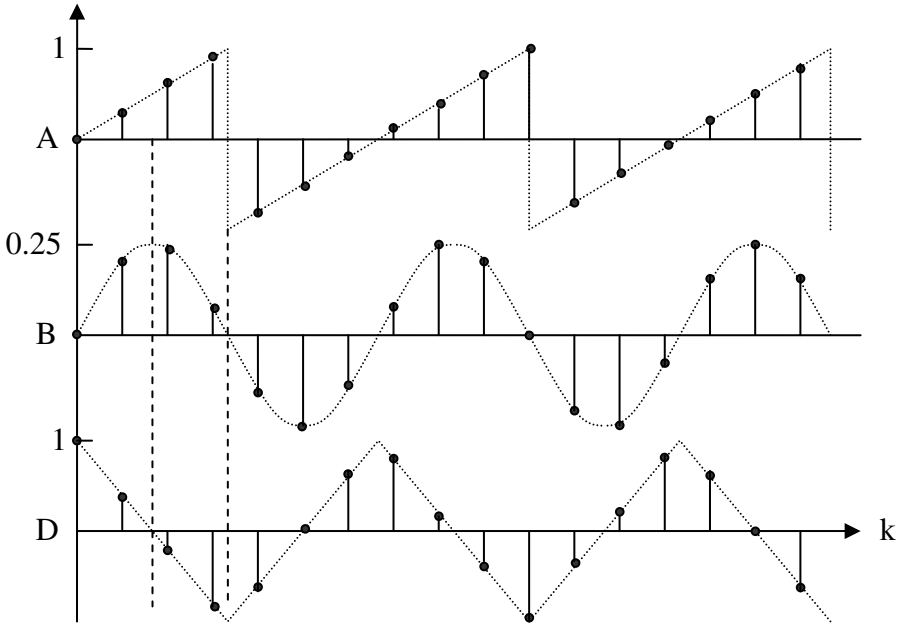


Fig. 77: Signals in the DPW Triangle Oscillator

Harmonic analysis of the parabolic segment train yields for the sampled values:

$$B[k] = \frac{8}{\pi^3} \sum_{n=1,3,5,..}^{\infty} \frac{1}{n^3} \sin(2\pi n f_o kT)$$

Due to its spectral properties, the signal B[k] can replace a sinusoid when purity is uncritical (e.g. in LFOs) or in sub oscillators as long as the fundamental remains below $f_s/20$.

The signal C[k] is obtained directly from B[k]:

$$C[k] = B[k] - B[k-1] = \frac{8}{\pi^3} \sum_{n=1,3,5,..}^{\infty} \frac{1}{n^3} [\sin(2\pi n f_o kT) - \sin(2\pi n f_o (k-1)T)]$$

$$C[k] = \frac{16}{\pi^3} \sum_{n=1,3,5,..}^{\infty} \frac{\sin(\pi n f_o T)}{n^3} \cos\left(2\pi n f_o \left(k - \frac{1}{2}\right) T\right)$$

For frequency components well below f_s, the condition $\pi n f_o T << 1$ holds and C[k] can be approximated as:

$$C[k] \approx \frac{16 f_o T}{\pi^2} \sum_{n=1,3,5,..}^{\infty} \frac{1}{n^2} \cos\left(2\pi n f_o \left(k - \frac{1}{2}\right) T\right)$$

The Fourier series of a discrete time triangle is: $$Tri[k] = \frac{8}{\pi^2} \sum_{n=1,3,5,..}^{\infty} \frac{1}{n^2} \cos(2\pi n f_o kT)$$

From the above we conclude the following:

1. The scaling factor becomes $f_s/(2f_o)$.
2. The output is time shifted by half a sampling period.
3. The relative amplitude deviation for a component at the frequency f is $\sin(\pi f/f_s)/(\pi f/f_s)$. Example: For f = 20 kHz and f_s = 48 kHz, an attenuation of 2.6 dB occurs. Hence, post filtering may be omitted.
4. Aliased components can be calculated directly using the fact that a discrete time sinusoidal has an f_s-periodic spectrum. Example: f_o = 5 kHz, f_s = 48 kHz. The first component that is aliased below f_o occurs at n = 9 because $|f_s - 9f_o| < f_o$.
5. The closer a component gets to a multiple of f_s, the weaker it becomes. This reduces aliasing to low frequencies and is a very desirable consequence of taking the time difference (which has the magnitude response $2\sin(\omega T/2)$).

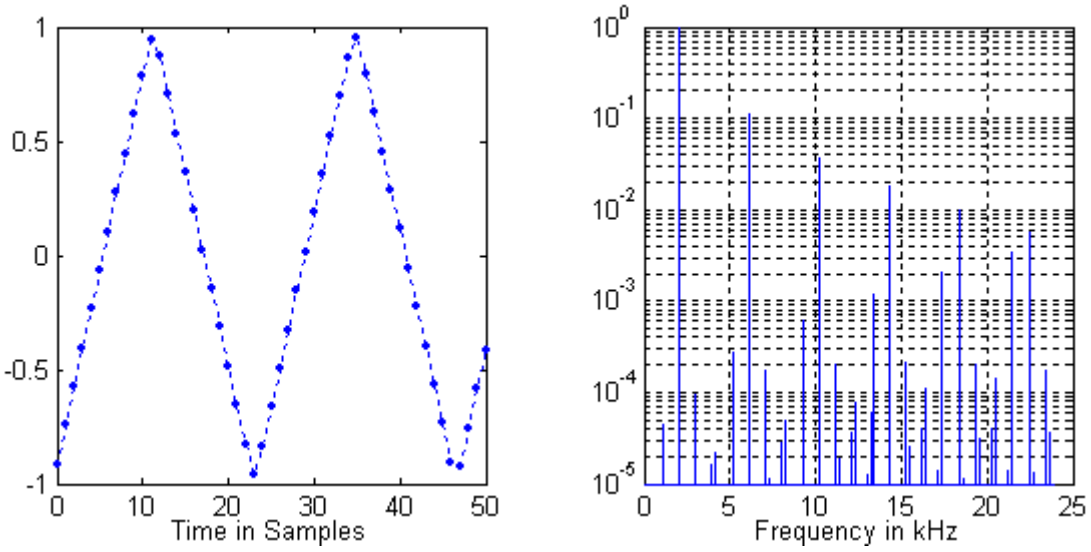


Fig. 78: Spectrum of the DPW Triangle Oscillator (f_o = 2.05 kHz, f_s = 48 kHz)

Aliased components one octave below the fundamental f_o are down by 80 dB for $f_o \approx 0.06 f_s$ and increase quickly to 63 dB for $f_o \approx 0.1 f_s$. Therefore, typical applications are sub oscillators and LFOs. On a double rate system, one might also consider using it as a main oscillator, at most with the top octave wrapped or replaced by a sinusoid. For this purpose, immediate phase control is feasible by calculating both B[k] and B[k+1] whenever the phase is set to a new value.

A sawtooth oscillator can be constructed following the same principle. Since the source signal is a train of non-alternating parabolic segments, its first derivative has discontinuities at the segment ends and the spectrum consequently declines only by $1/f^2$. This algorithm is recommended for audio purposes up to a fundamental frequency of $f_s/50$, which may be suitable for a fast LFO or a sub oscillator on a 96 kHz system.

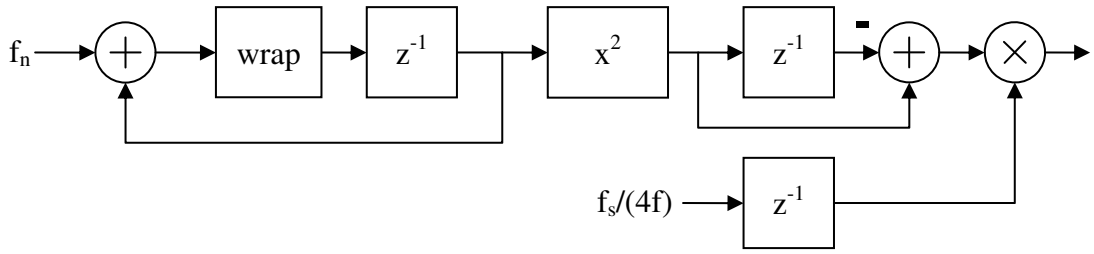


Fig. 79: DPW Sawtooth Oscillator

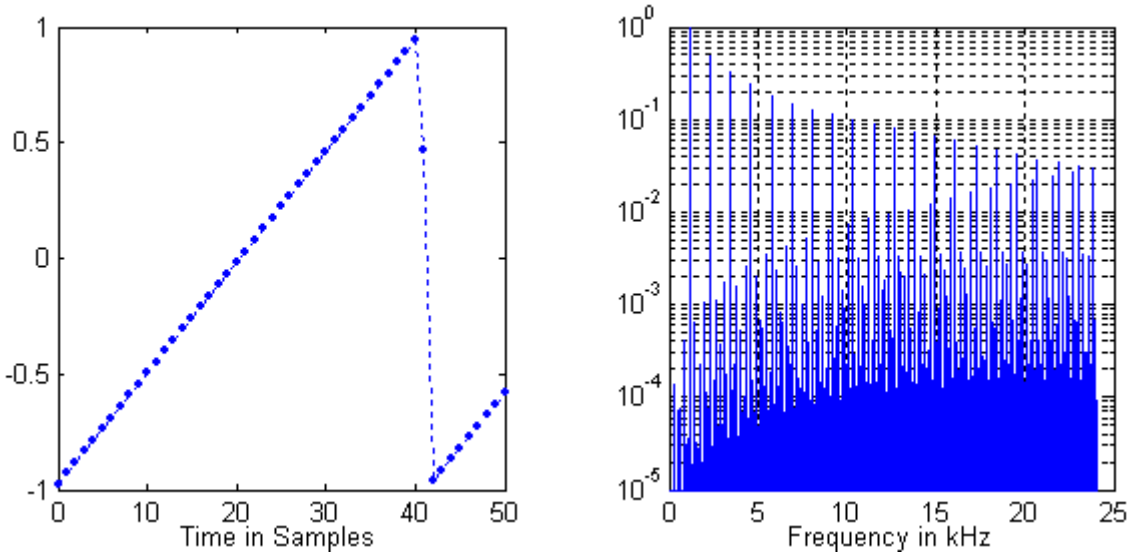


Fig. 80: Spectrum of the DPW Sawtooth Oscillator (f_o = 1.15 kHz, f_s = 48 kHz)

It should be mentioned that the very economical algorithm and its topology encourage oversampling (s. [15] for examples).

One may be tempted to extend the operating range using a smoother shaper function like $x\text{-}x^3$ instead of x^2 and then differentiate twice. This approach has severe drawbacks: High precision arithmetic is required, a one-sample spike occurs when the frequency is changed abruptly, the extension for immediate phase control is cumbersome, and finally it covers about half the frequency range of a BLIT-based oscillator with comparable computational effort. Therefore, we don't pursue this idea further. Instead, we are going to investigate the working principle with the intention to eliminate differentiators altogether and generalize the idea. It turns out that DPW oscillators can be viewed as an efficient realization of segment-based oscillators in which the segment is obtained by convolving the naïve waveform with rectangular pulses.

2.3.2 Segment-Based Integrated Prototype Signal (SIPS) Oscillator

First, we observe that in a DPW system the unit time delay z^{-1} can be replaced by a continuous time delay T without affecting the behaviour. Therefore, the discrete time system is equivalent to a continuous time model whose output is sampled at intervals T. For further analysis, we will treat it as continuous.

The goal is to render a trivial non-bandlimited prototype signal x(t) with minimal aliasing. At the input of the delayed difference block (Fig. 76, node B) we see the time integral X(t) of the prototype signal. X(t) equals the total area under the graph of x(t) (Fig. 81). The continuous output signal is the area difference within an interval T. In the discrete time system, the output is just this signal sampled in intervals of T.

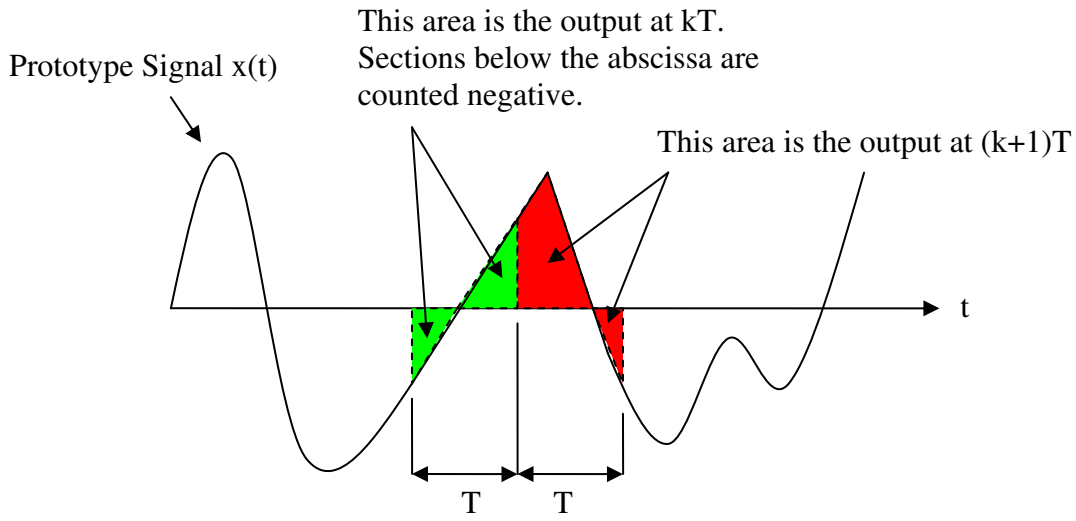


Fig. 81: Working Principle of Integrated Prototype Signal Oscillators

This process is equivalent to convolving the prototype signal x(t) with a rectangular pulse of width T followed by sampling at intervals of T. Consequently, the spectrum of x(t) is weighted by the spectrum $S_{rect}(f)$ of the pulse before the output signal is sampled.

Because

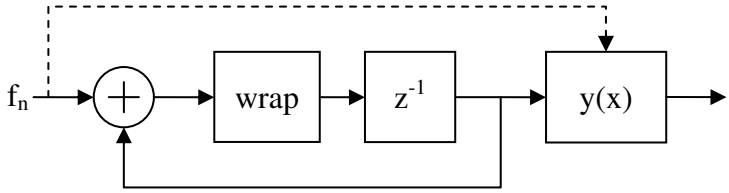
$$S_{rect}(f) = \frac{\sin(\pi f / f_s)}{(\pi f / f_s)}$$

exhibits zeros at integer multiples of the sample rate f_s, the output contains much less low frequency aliasing than a sampled prototype signal would.

It's important to see that two continuous time signals are convolved, so it's not the same as with the sample-based oscillator: Sliding a window over the prototype function and evaluating it in intervals of T would fail deplorably at discontinuities. The reason is that we would approximate the areas by single values obtained from sampling the prototype function at multiples of T around the current location. This is equivalent to sampling the prototype signal after it has been convolved with a discrete time version of the window, which does not have zeroes at multiples of f_s but a spectrum periodic in f_s.

From the above discussion we see, that the only thing we can do, is to find out how to calculate the continuous time convolution for a given prototype function.

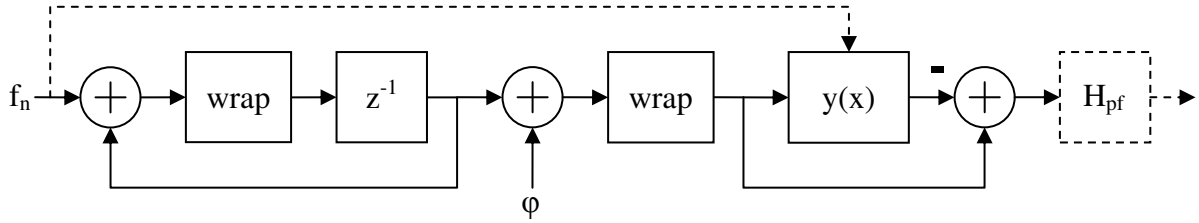
Fortunately, this is possible for the sawtooth and results in the system depicted in Fig. 82.



$$y(x) = \begin{cases} \operatorname{sgn}(x)(1-|x|)(f_s/f_o - 1) & ; |x| > 1 - f_o/f_s \\ x & ; \text{else} \end{cases}$$

Fig. 82: SIPS Sawtooth Oscillator

Since the SIPS sawtooth oscillator inherits the limited useful operating range from the DPW analogue, we might wish to go one step further and design higher order versions by repeated convolution of the sawtooth with the rectangular pulse. For convenience, we avoid nasty algebra, perform the convolution numerically with a technical computing tool, and exert a polynomial fit to the result. The 2nd order case yields a simple closed form solution (Fig. 83).



$$y(x) = \begin{cases} \operatorname{sgn}(x)[1 + (|x|-1)f_s/(2f_o)]^2 & ; |x| > 1 - 2f_o/f_s \\ 0 & ; \text{otherwise} \end{cases}$$

Fig. 83: Second Order SIPS Sawtooth Oscillator

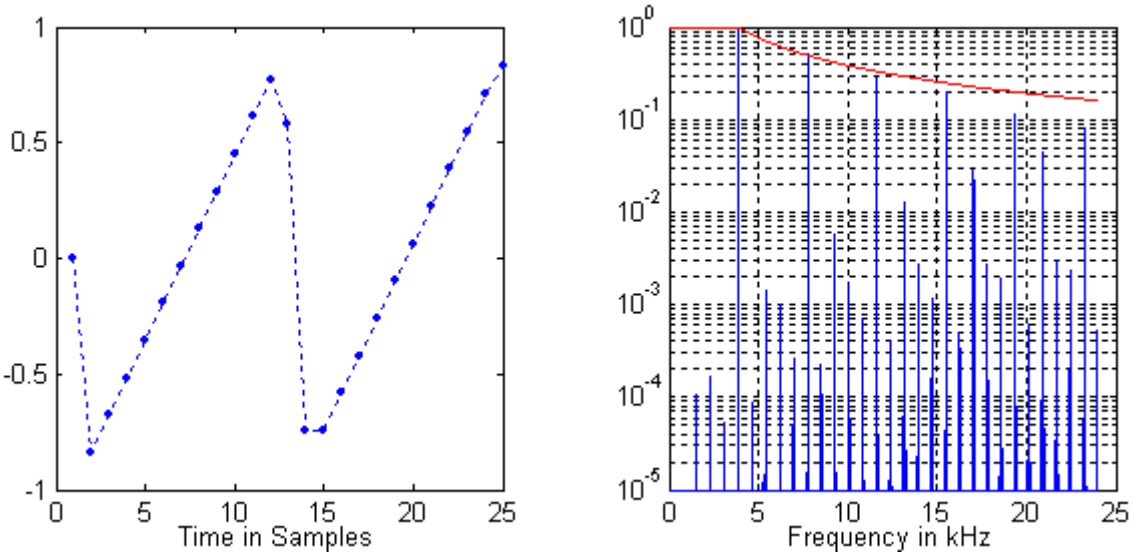


Fig. 84: Spectrum of the Second Order SIPS Sawtooth Oscillator
(f_o = 3.87 kHz, f_s = 48 kHz, no Postfilter, Red Line = Ideal Sawtooth Spectral Envelope)

This algorithm supports immediate phase control and performs well as a sub oscillator. We may even consider it as a main oscillator in 96 kHz systems. A postfilter H_{pf} could be inserted to equalize the $[\sin(\pi f/f_s)/(\pi f/f_s)]^2$ frequency response caused by the double convolution with the pulse:

$$H_{pf}(z) = \frac{1}{0.75 + 0.25z^{-1}}.$$

Higher orders lead to polynomials of at least 6th degree. In this case, the VA sawtooth oscillator is preferable since it has been tailored in the frequency domain for lowest audible aliasing.

We may notice that the function of the second order SIPS oscillator has a continuous first derivative at the segment boundaries. This is not as surprising as it seems: Any polynomial with the two lowest order coefficients being zero would satisfy this condition because the first derivative of the simple sawtooth is constant at those points. Unfortunately, this is no shortcut to better segments since a higher order polynomial from scratch namely produces the expected steeper spectral roll-off, but not necessarily the essential zeroes at or near multiples of the sample rate.

2.4 Oscillators based on Polynomial Shaping (PS)

2.4.1 Chebyshev Polynomial Shapers

Chebyshev polynomials of the 1st kind are defined by $\cos(n\varphi) = T_n(\cos\varphi)$. Feeding such a polynomial with a sinusoidal wave generates the pure n-th harmonic of the fundamental. If the source delivers a quadrature signal, it's also possible to produce orthogonal harmonics using Chebyshev polynomials of the 2nd kind, which are defined by $\sin(n\varphi) = \sin\varphi U_{n-1}(\cos\varphi)$.

n	$T_n(x)$	$U_{n-1}(x)$
1	x	1
2	$2x^2-1$	$2x$
3	$4x^3-3x$	$4x^2-1$
4	$8x^4-8x^2+1$	$8x^3-4x$
5	$16x^5-20x^3+5x$	$16x^4-12x^2+1$

For an output $y(t) = \sum_{n=1}^{5} A_n \cos(n\omega t)$ and $y_q(t) = \sum_{n=1}^{5} B_n \sin(n\omega t)$ the amplitudes can be packed into the coefficients of single polynomials: $P_Q(x) = \sum_{n=1}^{5} a_n x^n$ and $P_I(x) = \sum_{n=1}^{5} b_n x^n$.

n	a_n	b_n
1	$A_1-3A_3+5A_5$	$B_1-B_3+B_5$
2	$2A_2-8A_4$	$2B_2-4B_4$
3	$4A_3-20A_5$	$4B_3-12B_5$
4	$8A_4$	$8B_4$
5	$16A_5$	$16B_5$

Chebyshev series may also be evaluated efficiently by Clenshaw's recurrence.

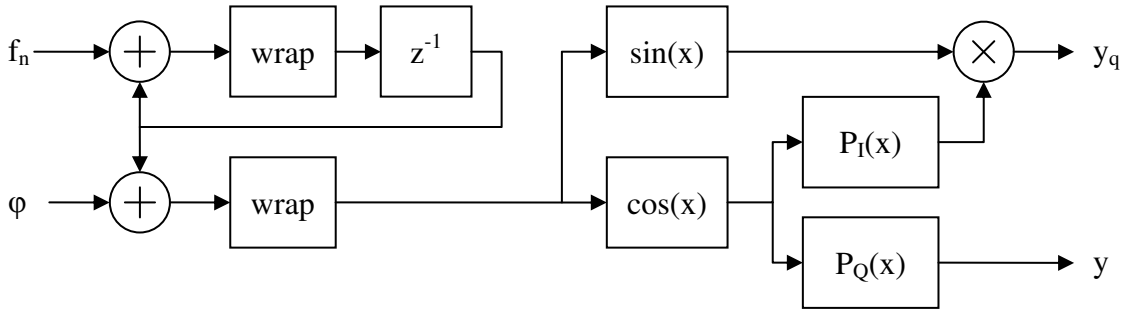


Fig. 85: CPS Oscillator

The CPS oscillator supports immediate phase control. If we only need 2nd and 3rd harmonics without quadrature output, replacing the cosine wave by a naïve triangle is an alternative. The following polynomials will make the modified oscillator sound good up to $f_o = f_s/12$:

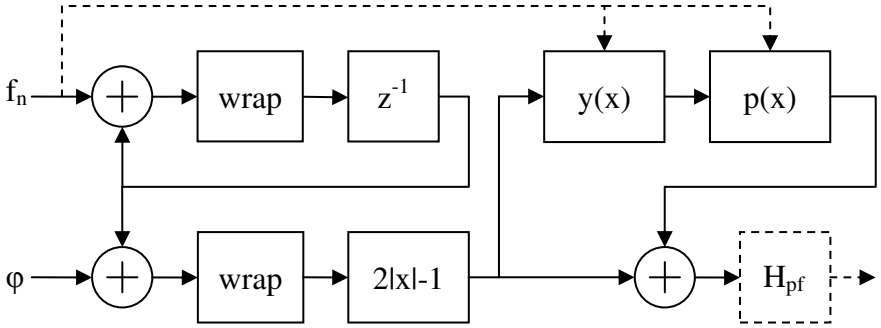
n	$T_{n(tri\_to\_cosine)}(x)$
1	$0.070726x^5 - 0.64089x^3 + 1.57007x$
2	$0.8989x^6 - 3.7774x^4 + 4.8750x^2 - 0.9980$
3	$4.7898x^7 - 15.5577x^5 + 16.3975x^3 - 4.6331x$

For n > 3, it's better to convert the triangle to a sinusoid and shape it afterwards.

2.4.2 Polynomial-Shaped Triangle Oscillator

In a completely different semi-empirical approach, we now try to replace the rather complex VA triangle oscillator. Aware that a bandlimited triangle can be formed from a simple one by subtracting a fixed length constant segment scaled proportionally to the fundamental frequency, we come to the following idea: Take a polynomial that converts a triangle into a sine and hope for the scaling to work satisfactorily. Such a design philosophy rarely yields professional results, but as the harmonics roll off quickly and the desired output is already sinusoidal at f_o = 6.6 kHz, we might once be lucky.

The ideal segment length turns out to be 3 in the main range, but then a sinusoid won't be generated below $f_s/3$ = 16 kHz, so the length is gradually changed in the transition region. Switching would also work if the amplitude is adjusted accordingly to avoid a discontinuity.



Frequency Update

$f_{lim} = 6f_o/f_s$
$r = \min(1, (f_{lim} + 16f_{lim}^{16})/2)$
$s = 1 - r$
$invs = 1/r$

Audio Update

$y(x) = \text{sgn}(x) \cdot invs \cdot (|x|-s)$; $|x| > s$
$y(x) = 0$; otherwise
$p(x) = (0.04575x^5 - 0.40889x^3)r$

$$H_{pf}(z) = \frac{1}{0.84 + 0.16z^{-1}}$$

Fig. 86: Bandlimited Wide Range PS Triangle Oscillator

The spectrum is quite clean and makes this oscillator a valuable alternative to the VA type. It can be expected to generate slightly more than half the processing load and delivers a sonically pleasant signal with aliasing unlikely to be heard. When customized appropriately, it performs excellent on a 96 kHz system.

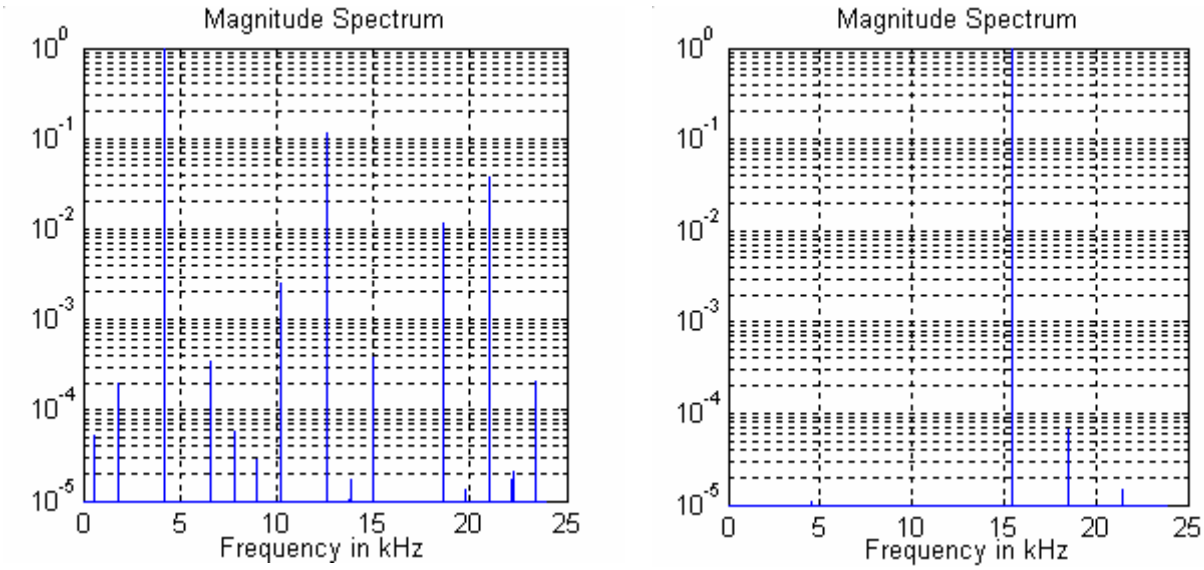


Fig. 87: Spectra of the Wide Range PS Triangle Oscillator
(f_o = 4.2 kHz and 15.5 kHz, f_s = 48 kHz)

2.4.3 Higher Function Shapers

Polynomial shaping of a sinusoid provides exact control over individual harmonics and is inherently band-limiting as the highest harmonic equals the degree of the polynomial. The overhead is rather high though. Sometimes, it's more desirable to have simple bandwidth control over a naturally evolving spectrum. Some window functions and their time integral perform very well as shapers in this case. An interesting choice is the Gaussian, which leads to a spectrum whose magnitude is described by modified Bessel functions of the 1st kind $I_n(x)$: It shows the steep decline of an FM spectrum without exhibiting dips. If we feed the Gaussian with a sine wave, a BLIT is produced:

$$x_{BLIT}(t) = e^{(\beta \sin(\pi f t))^2}$$; β = bandwidth control, f = fundamental frequency

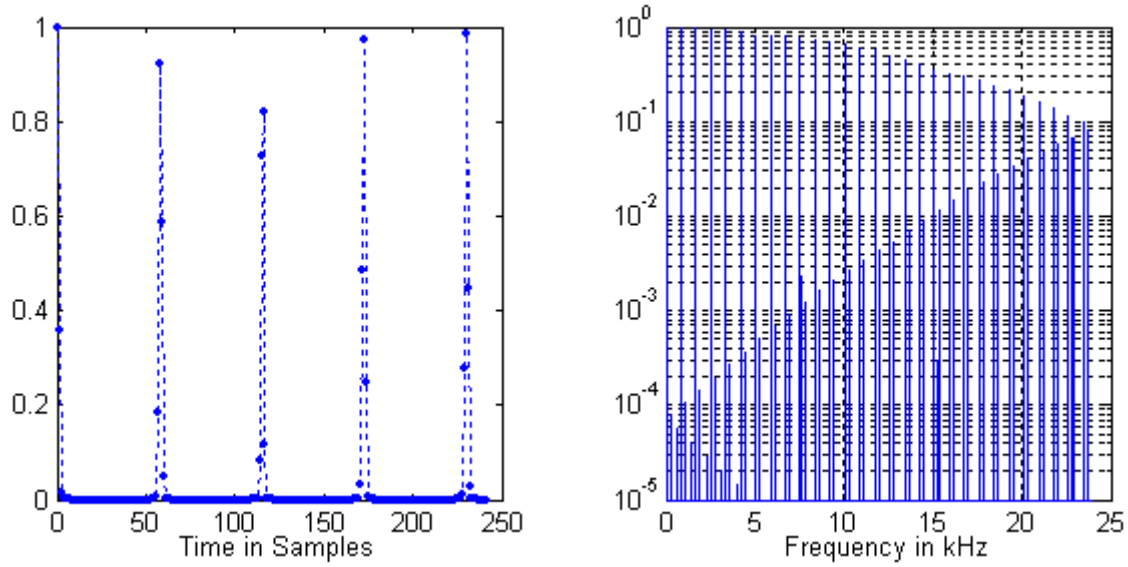
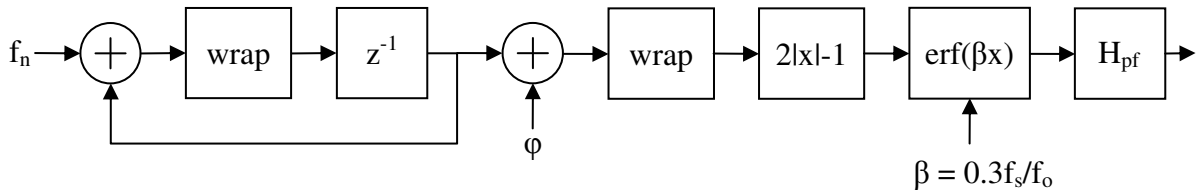


Fig. 88: Gaussian-Shaped Sinusoidal BLIT (discrete time version)

Spectrally, it is quite similar to the WSO-BLIT but with improved evenness, a slight advantage that might occasionally justify the appreciably higher resource requirements.

While bell-shaped functions are suitable to create an impulse train, sigmoidal shapes may be used for square wave generation. The VA pulse oscillator does of course fit that purpose too if the duty cycle is set to 50%, but this would be a sheer waste of processing time. Instead, a simple triangle is shaped by the error function erf(x), the time integral of the Gaussian, and β scales the input to adjust the bandwidth. This approach yields a high-quality bandlimited square wave using the embodiment of Fig. 89.



$$H_{pf}(z) = \frac{1}{0.64 + 0.36z^{-1}}$$

Fig. 89: Square Wave Oscillator using Error Function Shaped Triangle

The factor in the calculation of β may be tuned further by starting with 0.31 and linearly diminishing it to 0.28 between f_o = 2 to 4 kHz. The error function is preferably tabulated up to an argument of ±3 and set to ±1 outside this range.

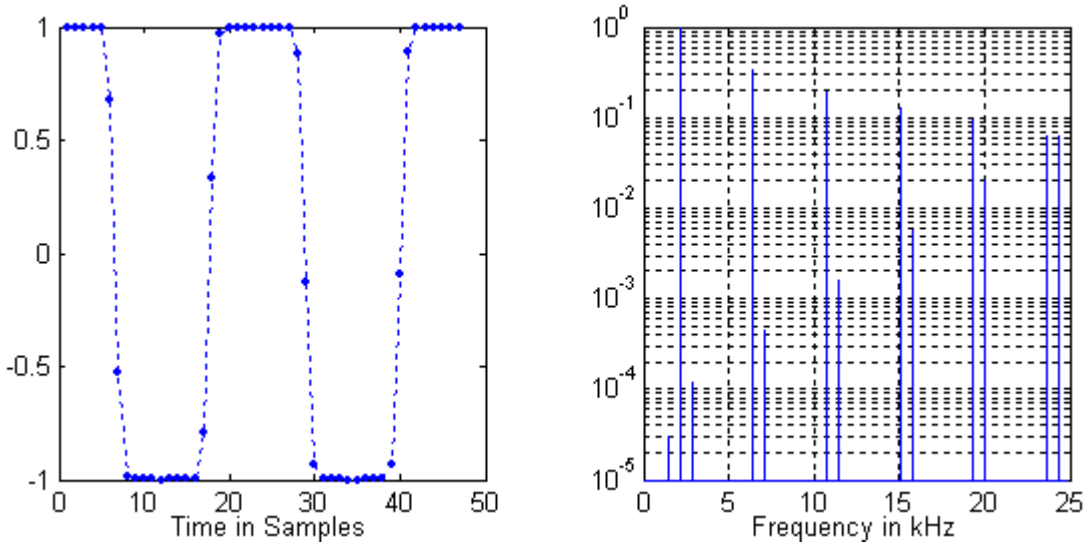


Fig. 90: Square Wave Oscillator Spectrum

This system works great in sub oscillator applications. If pulse width modulation is dispensable and switching to a sinusoidal for fundamentals above $f_s/6$ is implemented, it may even serve as a main oscillator. Furthermore, varying β enables filter-less bandwidth control.

We finish this section mentioning that several synthesis methods can be modelled as a special case of wave shaping. An example is FM synthesis, where the connection is uncovered by applying trigonometric addition theorems to a PM signal [8].

2.5 Phase Distortion (PD)

Any oscillator based on a phase accumulator can be extended with phase distortion (PD) capabilities. The idea is to shape the linearly increasing phase signal of the accumulator such that the original waveform of the oscillator is distorted in order to produce more harmonics within a well controlled band. In the Eighties, commercial PD synthesizers were built, although with moderate success, mainly because the competing FM synthesizers generated more complex spectra. With the revival of analog timbres and today's digital filters, there's good reason to reanimate this method.

Fig. 91 depicts the basic setup. f_{PD} is the shaper, preferably a sigmoid type function if we intend to modify an arbitrary waveform. f_{WV} translates a linearly increasing phase into the undistorted original waveform. Both functions assume an input range of [-1, 1] and deliver an output range of [-1, 1]. The complex arrangement to the right is just to cross fade from the distorted to the original phase when the modulation index m is below 1. In applications where it either always or never exceeds 1, the circuit can be simplified. Since PD potentially generates bias, a DC trap should follow the output.

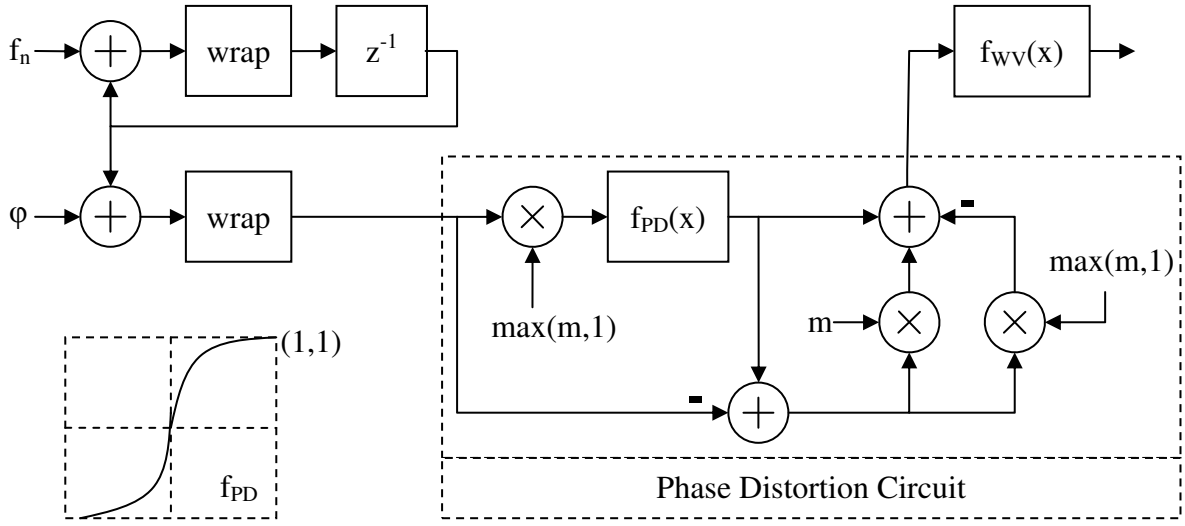


Fig. 91: Basic PD Setup

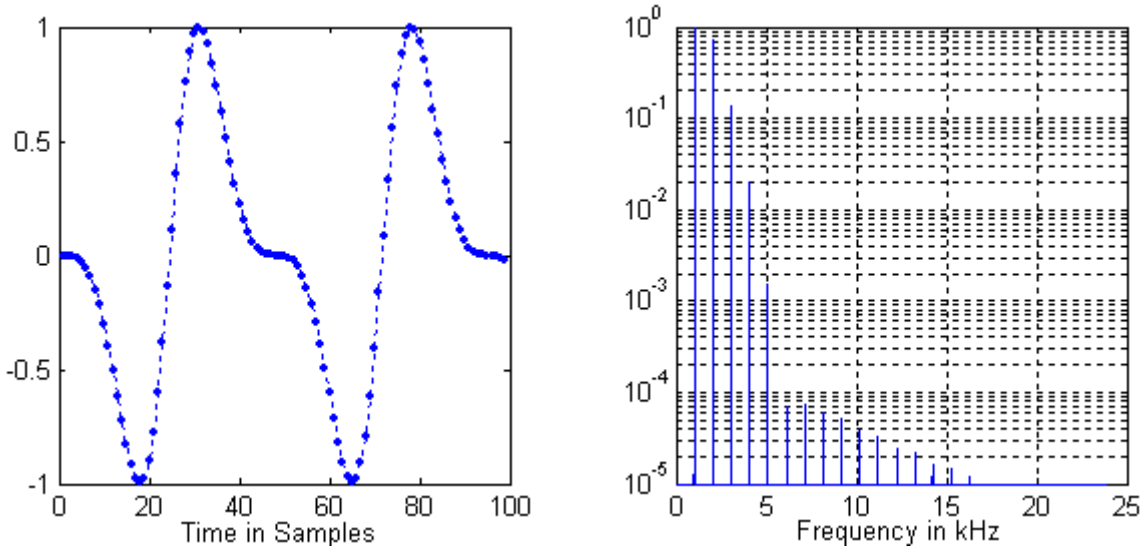


Fig. 92: Spectrum of the Basic PD Setup
($m = 1$, $f_{WV} = \sin(\pi x)$, $f_{PD} = 0.39675x^5 - 1.2935x^3 + 1.89675x$)

Sigmoidal functions allow parametric control of the slope while they remain bounded and saturate smoothly. The major criterion is spectral compactness since we want a lot of harmonics with minimal aliasing. Integrated window functions with zero 1st and small 2nd derivatives at the boundaries perform very well. Recommended shapers for sine and cosine waveforms are listed below.

$f_{PD}(x)$ (± 1 for $\lvert x\rvert > 1$)	$f_{WV} = \sin x$	$f_{WV} = \cos x$
$x + \sin(\pi x)/\pi$	+	+
$0.39675x^5 - 1.2935x^3 + 1.89675x$	+	+
$1.5x - 0.5x^3$	-	o

Some general statements can be made regarding these functions: It's less critical to shape a cosine because it already has zero 1st derivatives at the boundaries; the resulting low pass spectrum is however biased. A shaped sine acquires a band pass characteristic for high modulation indices but remains unbiased.

The input to the waveform function f_{PD} must span the interval [-1, 1] in order to complete a whole cycle. Hence, the modulation index m should be bounded to above 1 for the functions listed. We may cross-fade the shaped into the original phase signal for $m < 1$ and reduce m towards zero with increasing f_o to morph the oscillator output into a sinusoid for very high fundamental frequencies. A more sophisticated, although a bit luxurious, realization replaces $f_{WV}(x)$ by $f_{WV}(x)/f_{WV}(m)$.

When phase modulation is applied to the basic setup, FM spectra with enriched harmonics controlled by the PD modulation index are generated. Moreover, the annoying timbre of overdosed FM, which would be needed to create comparable harmonic content without PD, is completely absent.

A PD-BLIT is obtained by shaping a cosine and subtracting the bias. Compared to the WSO-BLIT, the postfilter must have about twice as much gain at 20 kHz, which is disadvantageous when the BLIT is modulated as the filter will then process a modified spectrum and its higher influence causes stronger spectral errors. However, this approach is recommended on double rate systems, where a simpler shaping function suffices and filter action is moderate.

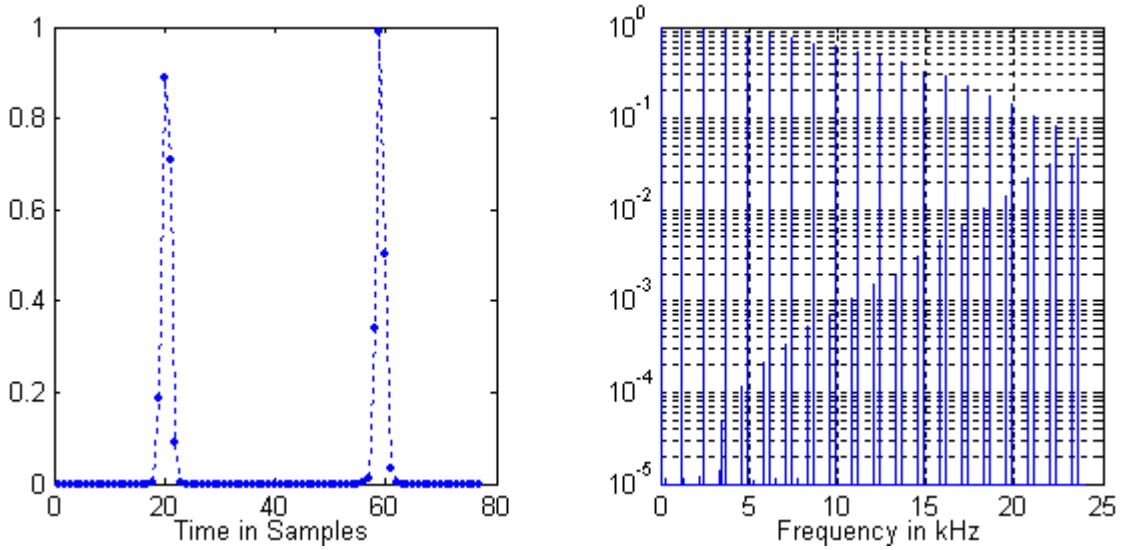


Fig. 93: Unfiltered PD-BLIT Spectrum
($f_{WV} = cos(\pi x)$, $f_{PD} = 0.39675x^5 - 1.2935x^3 + 1.89675x$)

2.6 Oscillators based on Windowed Segments

2.6.1 Principles

This versatile oscillator class occurs in a bewildering variety of embodiments whose suitability depends on the actual application and hardware infrastructure. However, the synthesis part is usually realizable with building blocks and techniques we already discussed.

The generic signal structure (Fig. 94) resembles those of a hard synced slave oscillator. The spectrum is the convolution of the spectra of a window function w(t) and an oscillating function s(t) sampled at multiples of the repetition rate f_o. In a discrete time realization, the segment start will usually not coincide with a sample point; something that is occasionally neglected for simplicity but has to be considered when the algorithm should sound good for fundamentals above a few 100 Hz.

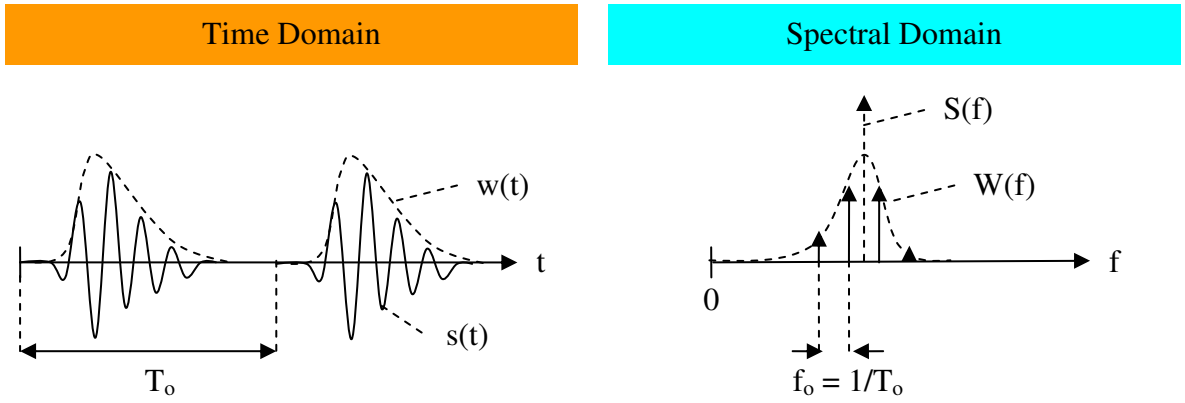


Fig. 94: Generic Windowed Segment Oscillator (WSO)

This setup can generate formants without using filters: The frequency of s(t) determines their centre, the width is inversely proportional to the length of w(t), and the fundamental frequency equals the repetition rate. Major criteria for the combination of w(t) and s(t) are smoothness with a corresponding steep spectral roll-off, finite support, parametric control, and low computational demand. Some classical algorithms, originally invented to model the human voice, are (continuous time prototype functions of a single cycle shown):

Method	s(t)	w(t)
VOSIM	$\sin^2(\omega_s t + \varphi)$	$\begin{cases} 0 & ;(t<0)\vee(t>N\tau) \\ \alpha^{floor[t/\tau]} & ;0 \le t \le N\tau \end{cases}$ $\tau = 1/(\pi\omega_s)$, N integer, $N\tau < T_o$, $0 \le \alpha \le 1$
FOF	$\sin(\omega_s t + \varphi)$	$\begin{cases} 0 & ;t \le 0 \\ \frac{1}{2}(1-\cos(\beta\pi t))e^{-\alpha t} & ;0 < t < 1/\beta \\ e^{-\alpha t} & ;t \ge 1/\beta \end{cases}$ $0 < \alpha, \beta$
Modified FOF	$\sin(\omega_s t + \varphi)$	$\begin{cases} 0 & ;(t \le 0)\vee(t \ge T_o) \\ \left(3(\beta t)^2 - 2(\beta t)^3\right)e^{-\alpha t} & ;0 < t < 1/\beta \\ e^{-\alpha t} & ;1/\beta \le t < (T_o - 1/\beta) \\ \left(3(\beta(T_o - t))^2 - 2(\beta(T_o - t))^3\right)e^{-\alpha t} & ;(T_o - 1/\beta) \le t < T_o \end{cases}$ $0 < \alpha, \beta$

2.6.2 VOSIM

Let's look at a VOSIM [9] example to get familiar with windowed segment oscillators. Typical VOSIM spectra have a formant peak and a strong cluster around the fundamental that adds to the body of the sound. The special case of two adjacent full scale cycles leads to a particularly efficient realization of a bandlimited formant oscillator. For fundamentals above $f_s/12$, the output fades into a sinusoid in the system shown (Fig. 95).

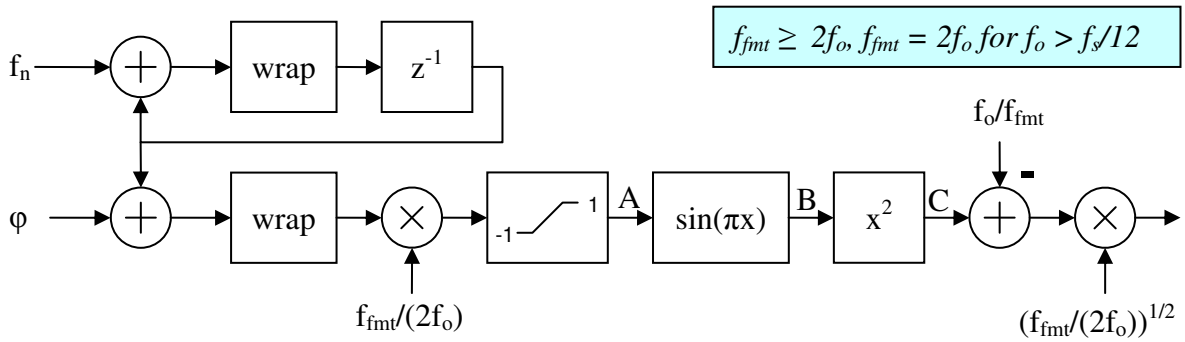


Fig. 95: Basic VOSIM Formant Oscillator

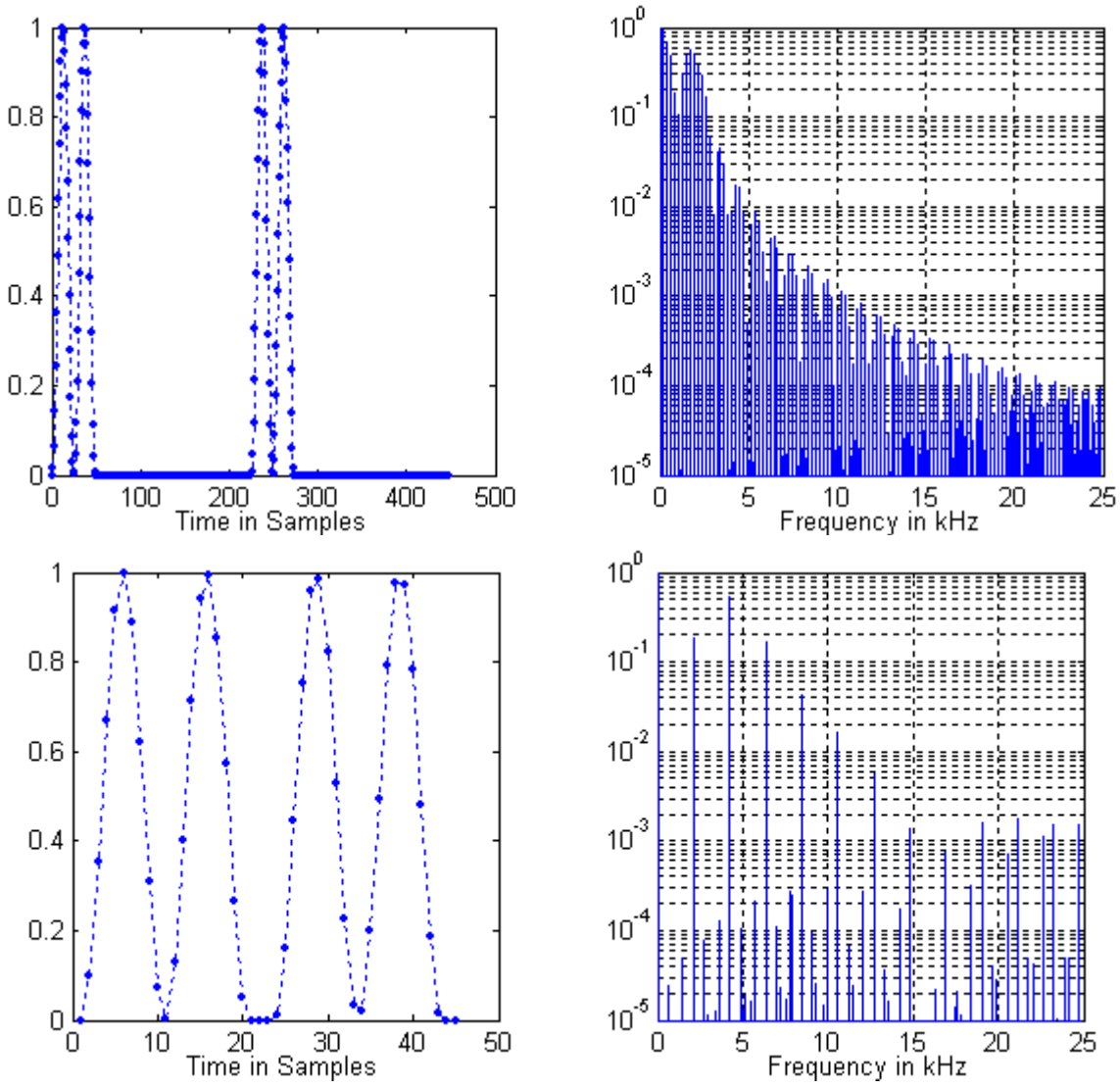


Fig. 96: Basic VOSIM Formant Oscillator Spectra (at node C)

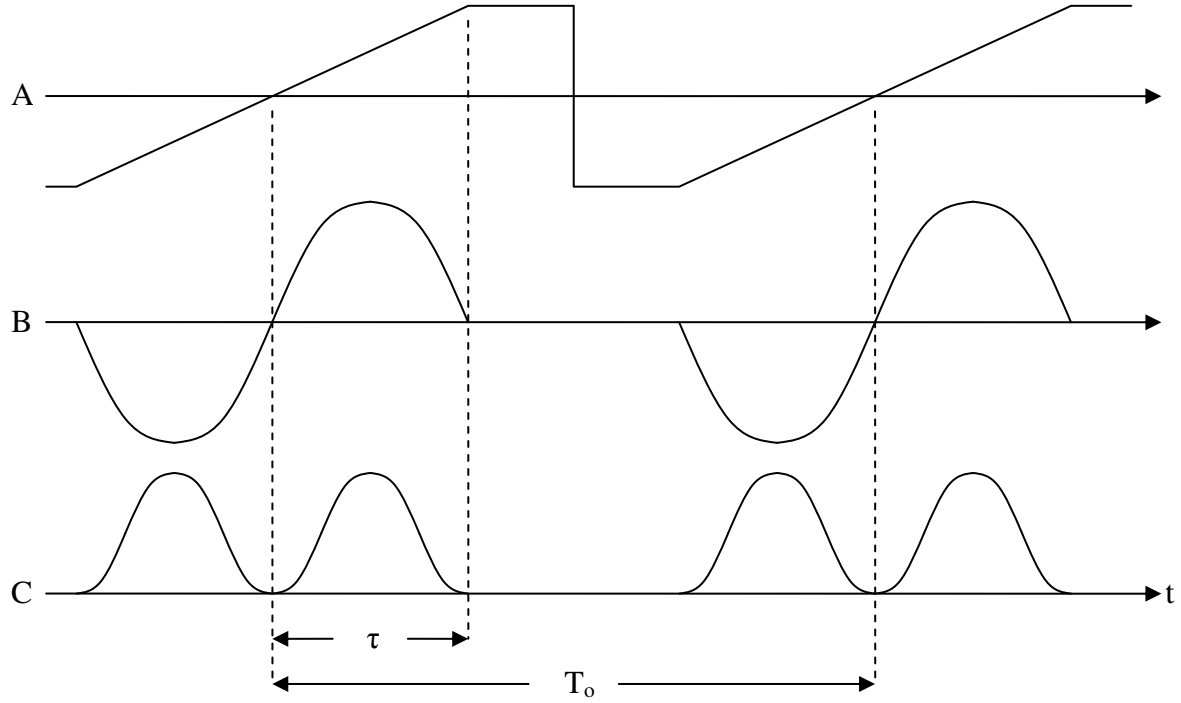


Fig. 97: Continuous Time Prototype Signals in the Basic VOSIM Formant Oscillator

Within the system's operating range, the spectra of the continuous time prototype signal and the discretized version barely differ. We calculate the Fourier series with τ denoting the width of a single raised cosine segment to obtain the amplitudes A_n of the harmonics at node C in Fig. 95:

$$A_n = 2|a_n| = \left| \frac{2}{T_o} \int_{-\tau}^{\tau} \left[\frac{1-\cos(2\pi t/\tau)}{2} \right] \cos(2\pi n t/T_o)\, dt \right| = \left| \frac{\sin(2\pi n \tau/T_o)}{\pi n \left[1-(n\tau/T_o)^2 \right]} \right| \qquad ; n \geq 1$$

$$A_o = \tau/T_o$$

Starting at the low end, the spectrum exhibits a dip at $n \approx T_o/(2\tau)$, peaks for $n \approx T_o/\tau$ with a value $A_n \approx \tau/(2T_o)$, and then asymptotically decays at -18 dB/octave. The approximate formant frequency f_{fmt} is linked to τ by the relation $f_{fmt} = nf_o = 1/\tau$. Bias compensation is realized by subtracting $A_o = \tau/T_o = f_o/f_{fmt}$ from the output. We deduce from the time domain that the signal energy is proportional to τ/T_o and therefore also to f_o/f_{fmt} for $2\tau < T_o$. This leads us to the employed normalization scheme.

When synchronous VOSIM signals with identical fundamentals are superimposed, the fundamental regions add up in phase. The same also holds for adjacent formant components. In practical applications, it's often a good idea to invert one of the VOSIM signals. While such a system sounds pleasant and is real fun to experiment with, we should be aware of alternatives that provide easier handling of spectral control, a feature coming to great advantage in the precise emulation of the human voice and other formant-based instruments.

In the next two subsections, we are going to follow up the idea of combining two simple segments and see what we get.

2.6.3 Modified VOSIM Formant Train Oscillator

An unbiased formant train without prominent fundamental region is obtained by inverting one of the cosine segments. It sounds rich and features the interesting characteristic of a damped pipe resonator. Unfortunately, the broad spectrum also makes it prone to aliasing, thus limiting the recommended formant center frequency to about 2.5 kHz on a 48 kHz system. However, we may replace band-limited impulses of section 1.3 for the raised cosines to extend both formant and fundamental range to $f_s/8$. Another option is to superimpose two time-shifted BLITs of section 2.6.6.

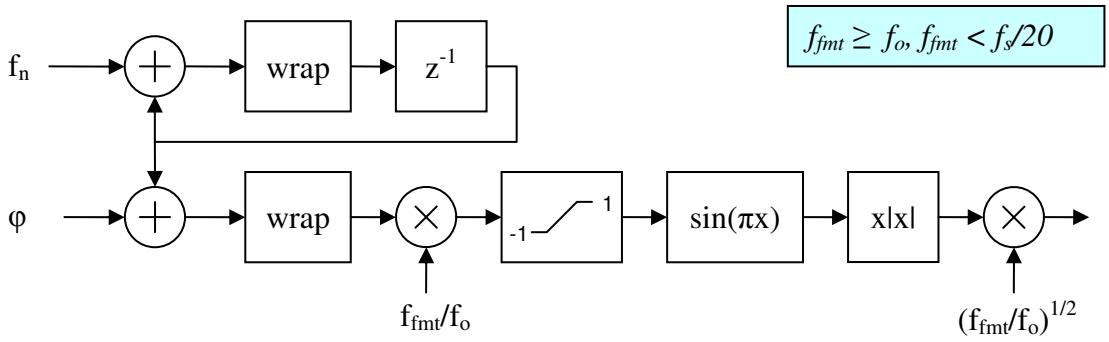


Fig. 98: Modified VOSIM Formant Train Oscillator
(Recommended for double rate systems)

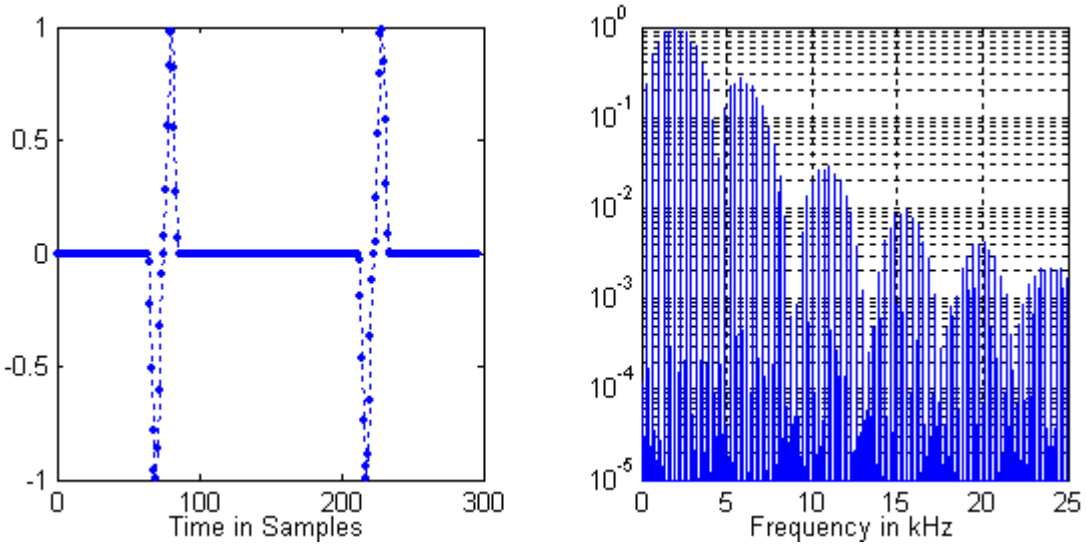


Fig. 99: Spectrum of the Formant Train Oscillator (not normalized)

2.6.4 Wide Formant Oscillator (WFO)

We may wonder if it's possible to further modify the preceding algorithm to render it useable over the whole audio range. The crux is the raised cosine, which is just too narrow when applied to every single segment. A way to stretch it without affecting the smooth borders and the shape itself too much is to weight a single sine cycle with a raised cosine that extends over the whole cycle:

$$y_{sgm}(t) = \begin{cases} \frac{8}{3\sqrt{3}} \sin(2\pi f_{fmt} t) \cos^2(\pi f_{fmt} t) & ;|t| < 1/(2 f_{fmt}) \\ 0 & ;otherwise \end{cases}$$

It turns out that this segment delivers a general purpose band pass spectrum when repeated. A polynomial approximation may be used in the actual wide formant oscillator:

$$y_{sgm}(x) \approx P(x) = -2.4561x^{11} + 12.5485x^{9} - 27.8871x^{7} + 32.7775x^{5} - 19.8169x^{3} + 4.8341x$$

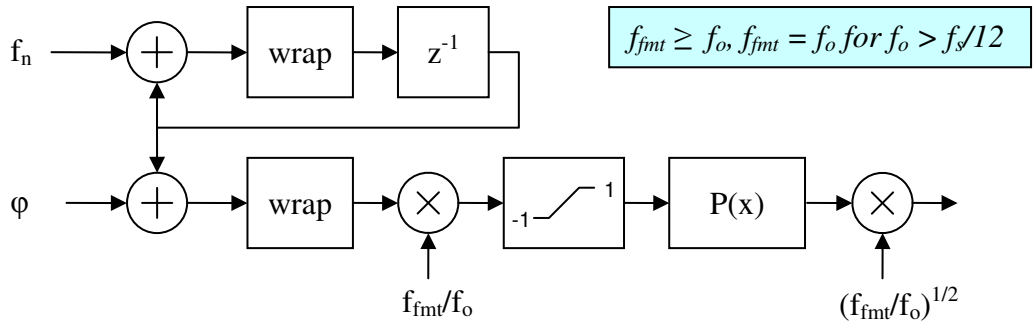


Fig. 100: Wide Formant Oscillator (WFO)

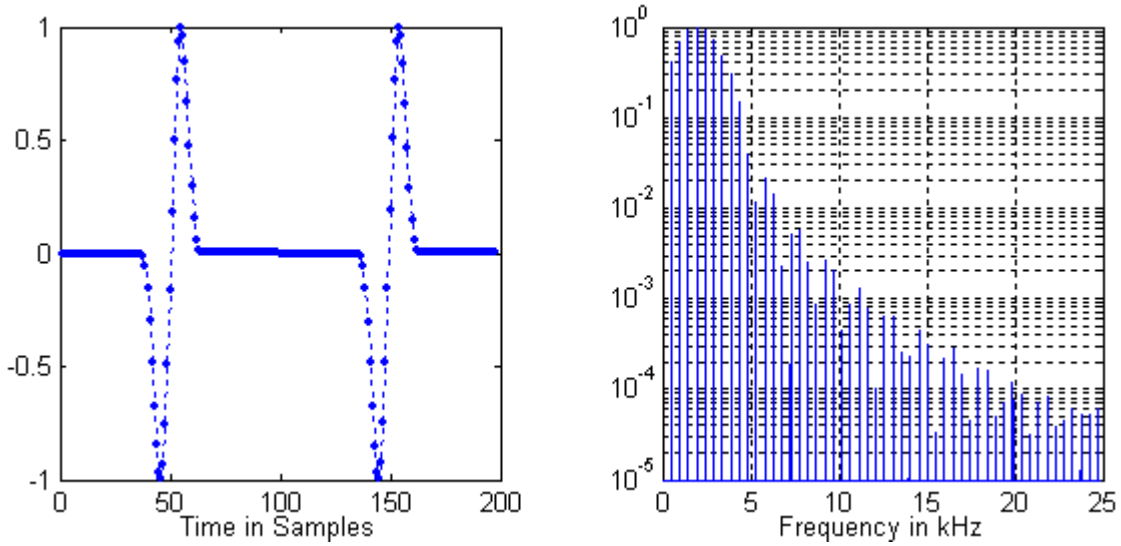


Fig. 101: WFO Spectrum (not normalized)

For fundamental frequencies above $f_s/12$, the output consists of the fundamental and the 1st harmonic at half the amplitude.

2.6.5 Variable Width Formant Oscillator (VWFO)

Formant synthesis not only requires parametric width control but also much narrower peaks than those provided by the WFO. This is easily achieved by varying the width of the raised cosine window. In the system of Fig. 102, β sets the width relative to f_o.

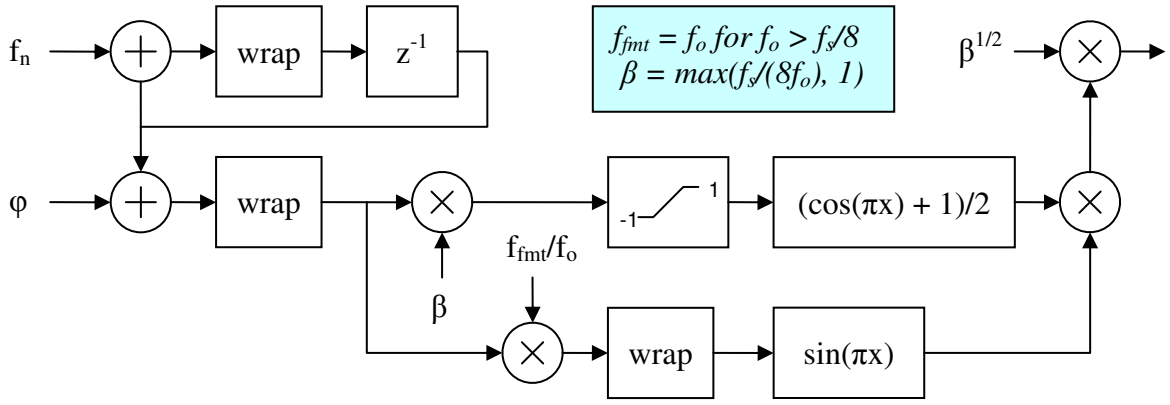


Fig. 102: Variable Width Formant Oscillator (VWFO)

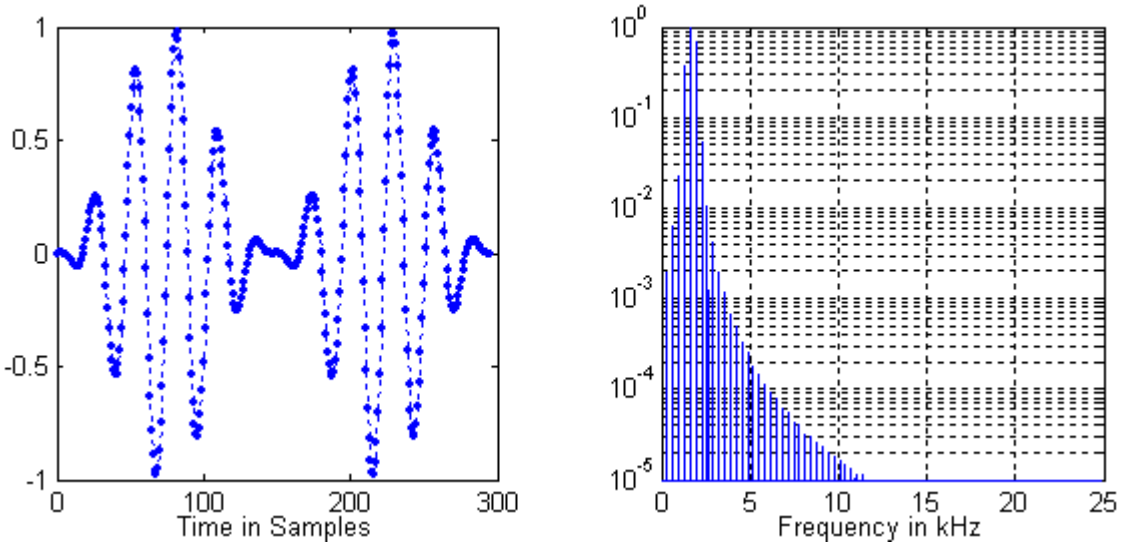


Fig. 103: VWFO Spectrum (β = 1, not normalized)

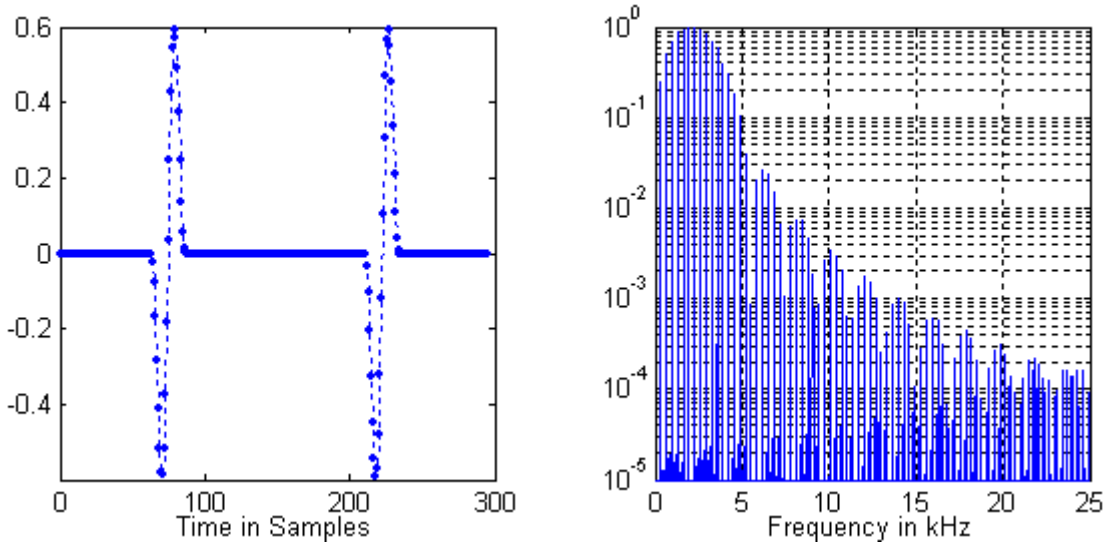


Fig. 104: VWFO Spectrum (β = 6, not normalized)

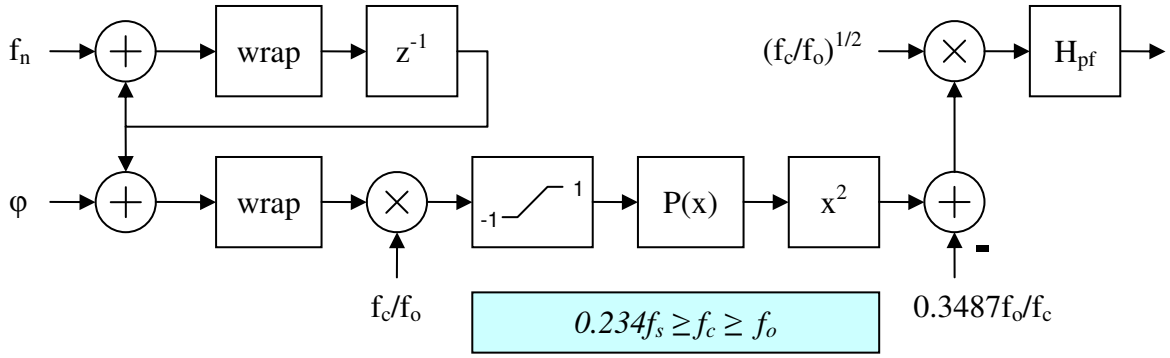
Minimum formant peak width occurs for $\beta = 1$ and is determined by the spectrum of the raised cosine. The resulting –6 dB bandwidth becomes $BW_{-6dB} \approx 2\beta f_o$ (approximately, because the spectrum is symmetrical to the y-axis but with opposite signs, hence superimposed components from one side partially cancel those of the other).

For $f_o > f_s/8$, the output of consists of the fundamental and the 1st harmonic at half the amplitude.

2.6.6 WSO-BLIT Oscillator

Windowed segment oscillators also make good variable bandwidth BLIT sources if an appropriate bandlimited impulse segment is used. When VOSIM is configured to repeat a single fixed width raised cosine segment, the spectrum can be made flat up to $f_s/8$ without introducing audible aliasing. This is ideal for voice synthesis where too much high frequency content is perceived as unwanted buzz.

Full audio bandwidth BLITs are obtained by periodically repeating one of the bandlimited impulses of section 1.3. At least 10 kWords of memory are recommended to tabulate the segment for non-interpolated readout. Alternatively, the table look-up can be replaced by a polynomial approximation. Large impulses will limit the maximum fundamental frequency and lead to polynomials with over a dozen terms. Therefore, we stick to a short 4-sample impulse and get the system shown in Fig. 105 with $f_{o(max)} = f_s/4$ and ± 0.8 dB ripple in the spectrum. Approximating the square root instead of the segment itself saves two product terms.



$$P(x) = -0.2139x^{10} + 1.0583x^{8} - 2.5894x^{6} + 3.7647x^{4} - 3.0185x^{2} + 1$$

$$H_{pf}(z) = \frac{1}{0.6 + 0.4z^{-1}}$$

Fig. 105: WSO-BLIT Oscillator

If two time-shifted WSO-BLITs are added or subtracted, a variety of interesting spectra including wide range versions of the basic VOSIM formant and the formant train oscillators are obtained. Additionally, pulse width modulation is achieved by varying the time shift.

In general, the raised cosine may be replaced by a tailored bandlimited impulse in each of the aforementioned designs in order to roughly double the useful frequency range.

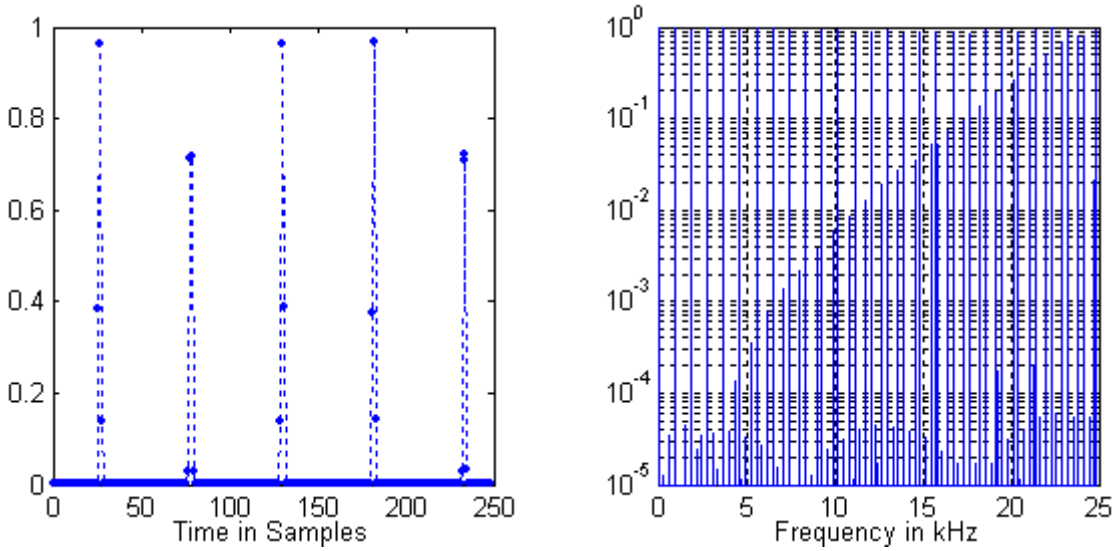


Fig. 106: WSO-BLIT Spectrum ($f_c = 0.234f_s$, including bias, not normalized)

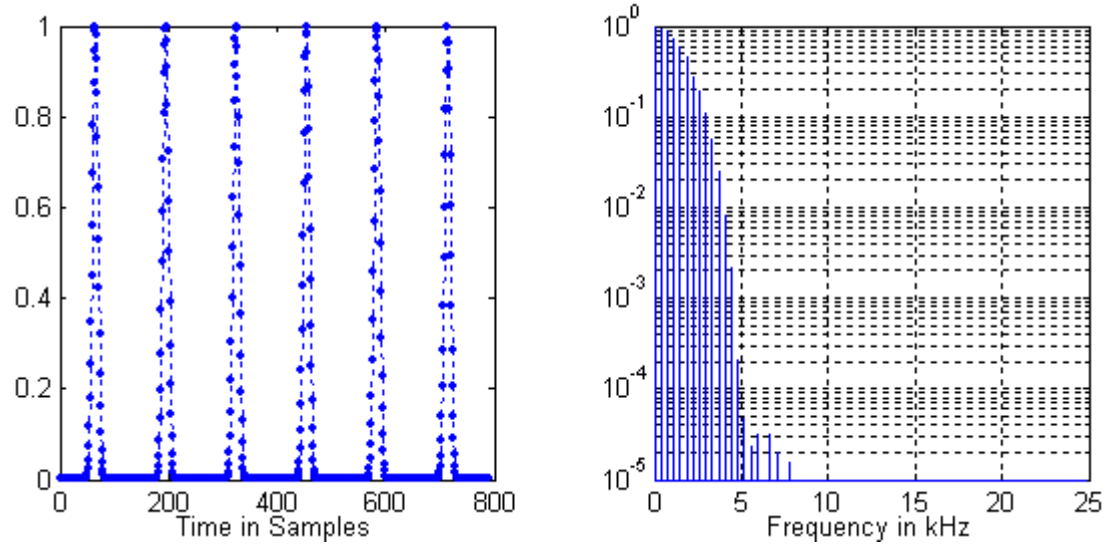


Fig. 107: WSO-BLIT Spectrum ($f_c = 0.026f_s$, including bias, not normalized)

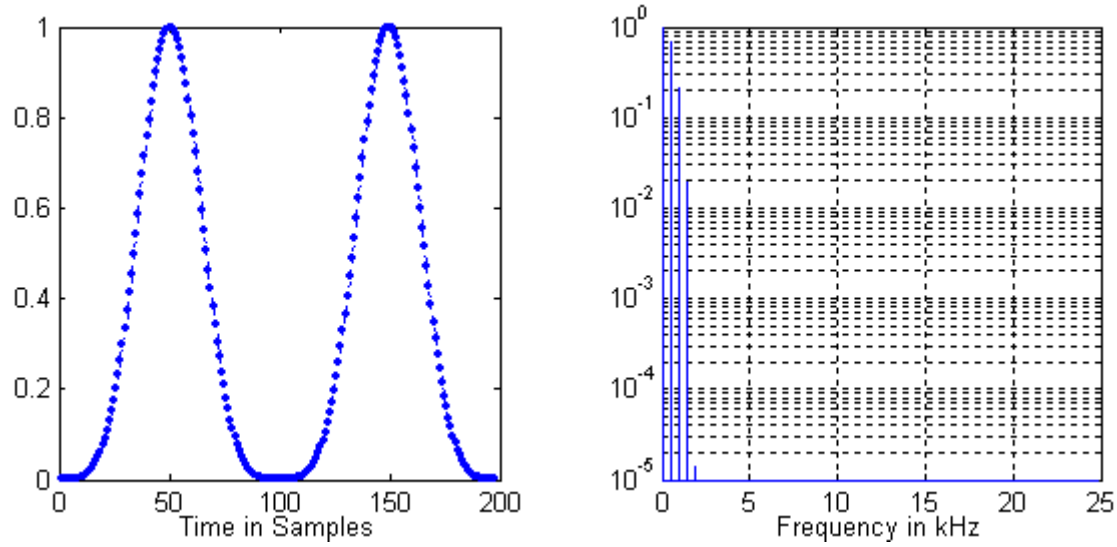


Fig. 108: WSO-BLIT Spectrum ($f_c = f_o$, including bias, not normalized)

2.6.7 FOF

FOF (fonction d'onde formantique) generates a band pass spectrum suitable for additive formant synthesis. It outperforms VOSIM when straightforward control of the spectrum and the ability to create narrow peaks are crucial. The latter is achieved at the expense of a long-tailed (theoretically infinite) signal which in practice causes the computational effort to depend on the parameters. FOF is a key ingredient of CHANT [10], a program for the synthesis of the singing voice and other instruments.

Modified FOF has finite support and uses a polynomial that can be evaluated faster than the cosine on today's machines. Due to finite segment length, the minimum formant width is limited by the fundamental.

In both methods, a DC trap should be added to eliminate any residual output bias.

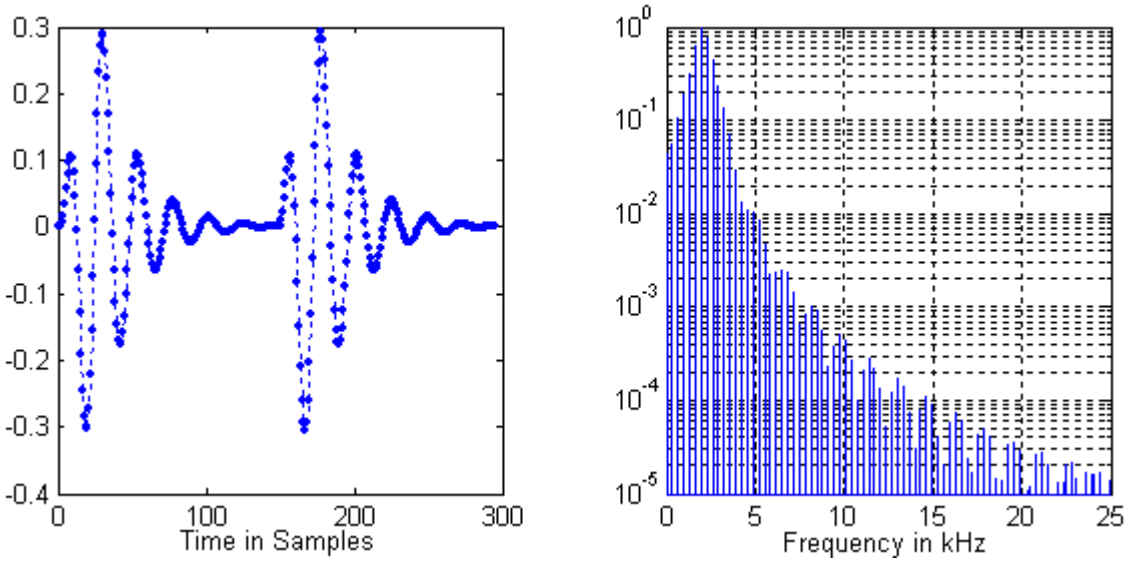


Fig. 109: Modified FOF Spectrum

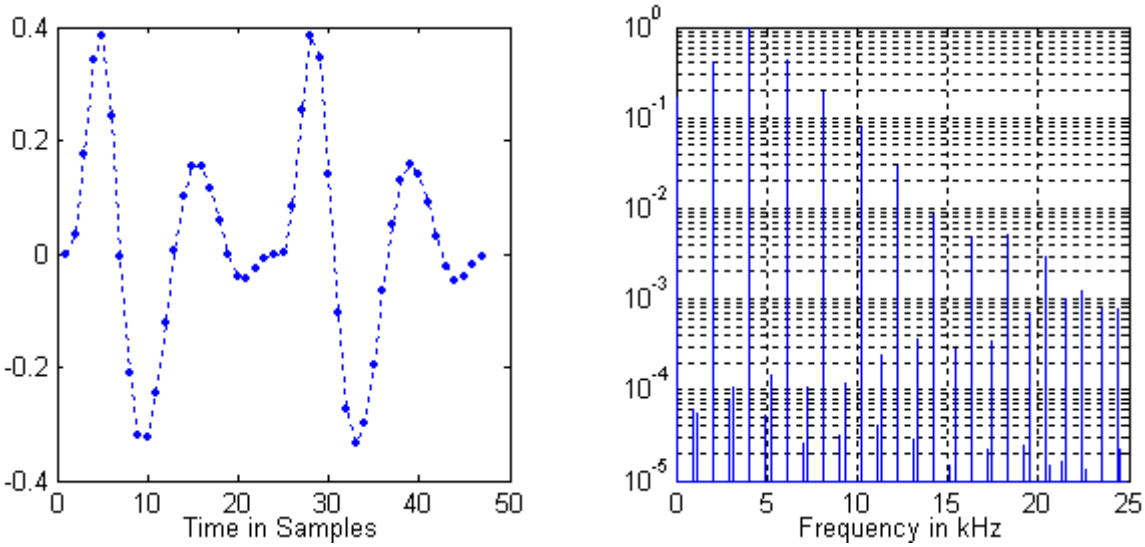


Fig. 110: Modified FOF Spectrum

2.6.8 Remarks on Windowed Segment Oscillators and Formants

If the fundamental is bounded to the lower decades of the audio spectrum, one may consider second order feedback sinusoidal oscillators to compute trigonometric functions as the laborious setting of initial conditions occurs rarely and they can be tweaked for exponential decay.

Some sophisticated algorithms have been developed and optimized to not produce spectral dips when formants overlap (e.g. PAF, IRCAM, 95). When it comes to just adding a vowelish timbre with low control overhead, delay-based comb filters are a great sounding option, because they generate a formant train which is characteristic of many natural resonators. Some ad hoc methods yield likewise spectra on commercial synthesizers, for example: FM (often combined with Hard Sync), audio rate sawtooth modulation of the filter cut-off frequency, nonlinear distortion of a band pass signal, vocoder techniques like feeding a 4^{th} order band pass filter bank with a broadband spectrum.

So far, the segments have been parametric functions. Interesting applications like formant preserving pitch shift arise from using samples or real-time audio as segments. Professional systems of this kind tend to be very sophisticated and include a lot of signal analysis, so they are not covered in the oscillator chapter. However, the main idea is simple enough to start experimenting:

- Grab a section of a time-domain signal, weight it with a window function, and repeat it at the desired fundamental frequency with or without overlap avoiding truncation.
- In the frequency domain, the following happens: The spectrum of the selected section is convolved with the spectrum of the window and then discretized to multiples of the new fundamental frequency.

The section length, and to a lesser degree the window type, are critical. For oscillator purposes, a raised cosine window with a length of one period of the original signal is suggested. Although there are elaborate methods for period estimation, simple autocorrelation algorithms perform well enough to take first steps on tuned signals with a prominent harmonic structure. A more advanced approach is described in [12].

Granular synthesis is a superset of segment-based oscillators with the extension that not only the segments (called “grains”) but also their time of occurrence is arbitrary. This way, harmonic as well as noisy and intermediate phenotypes are generated and morphed interactively into each other. It should be mentioned that a Gaussian is often preferred to trigonometric windows in this application.

Appendix A: Oscillator Selection Guide

Type	Section	Immediate Phase Control	Immediate Frequency Control	Syncable	1/f Tracking required	Recommended Fundamental Frequency Range @ f_s = 48 kHz (f_o/f_s)	Useful Fundamental Frequency Range @ f_s = 48 kHz (f_o/f_s)	Postfilter / DC Trap required	FM / PM via Frequency In	FM / PM via Phase In	Spectral and Temporal Fidelity	Processor Cycles	Conditional Instructions @ f_s	Wrap Around	Memory (kWords)
Sample-based	1.4	+	+	+	N	0.46	0.46	N	+	+	+	o	N	N	>100
VA Sawtooth + Pulse	1.5	+	+	+	Y	0.25	0.33	Y	+	+	+	+	Y	Y	6
Dito, without table	1.5	+	+	+	Y	0.25	0.33	Y	+	+	+	o	Y	Y	0
VA Triangle	1.5	+	+	+	Y	0.25	0.33	Y	+	+	+	o	Y	Y	1
Dito, without table	1.5	+	+	+	Y	0.25	0.33	Y	+	+	+	-	Y	Y	0
Sinusoidal FM/PM	1.7, 1.9	+	+	+	N	0.46	0.46	N	+	+	+	+	N	Y	0
Dito, with Feedback	1.9.2	-	+	-	N	0.46	0.46	Y	+	+	+	+	N	Y	0
Dito, Sawtooth	1.9.3	-	+	-	N	0.46	0.46	Y	+	+	o	o	N	Y	0
Dito, Triangle	1.9.3	-	+	-	N	0.46	0.46	Y	+	+	o	+	N	Y	0
Wavetable	1.10	+	+	+	Y	0.46	0.46	N	+	+	+	+	N	Y	6..15
CF Sinusoidal	2.1.1	o	+	-	N	0.46	0.46	N	o	-	+	++	N	N	0
DF1 Sinusoidal	2.1.2	o	o	-	N	0.46	0.46	N	-	-	+	++	N	N	0
Chamberlin Sinusoidal	2.1.3	o	o	-	N	0.46	0.46	N	o	-	+	++	N	N	0
DSF-BLIT	2.2	+	+	+	N	0.46	0.46	N	+	+	+	-	N	Y	0..30
Windowed DSF-BLIT	2.2	+	+	+	Y	0.12	0.20	N	+	+	+	o	Y	Y	1..30
DSF-BLIT Sawtooth	2.2	-	+	-	Y	0.46	0.46	N	+	o	+	-	N	Y	0..30
DPW, Sawtooth	2.3.1	o	+	o	Y	0.02	0.03	Y	+	o	+	++	N	Y	0
DPW, Triangle	2.3.1	o	+	o	Y	0.08	0.12	Y	+	o	+	++	N	Y	0
SIPS, Sawtooth	2.3.2	+	+	+	Y	0.02	0.03	Y	+	+	+	+	Y	Y	0
SIPS2, Sawtooth	2.3.2	+	+	+	Y	0.08	0.12	Y	+	+	+	+	Y	Y	0
CPS	2.4.1	+	+	+	N	0.46	0.46	N	+	+	+	o	N	Y	0
PS, Triangle	2.4.2	+	+	+	Y	0.46	0.46	Y	+	+	+	+	N	Y	0
PS, Square	2.4.3	+	+	+	Y	0.14	0.17	Y	+	+	+	+	N	Y	8
PD-BLIT	2.5	+	+	+	Y	0.46	0.46	Y	+	+	o	+	Y	Y	0
VOSIM, Formant	2.6.2	+	+	+	Y	0.23	0.23	N	+	+	o	+	Y	Y	0
Formant Train	2.6.3	+	+	+	Y	0.04	0.05	N	+	+	+	+	Y	Y	0
WFO	2.6.4	+	+	+	Y	0.23	0.23	N	+	+	+	+	Y	Y	0
VWFO	2.6.5	+	+	+	Y	0.08	0.23	N	+	+	+	o	Y	Y	0
WSO-BLIT	2.6.6	+	+	+	Y	0.23	0.23	Y	+	+	+	+	Y	Y	0
FOF, unlimited	2.6.7	+	+	-	N	0.46	0.46	Y	+	+	+	-	Y	Y	0
FOF, truncated	2.6.7	+	+	+	N	0.04	0.08	Y	+	+	+	o	Y	Y	0
FOF, modified	2.6.7	+	+	+	N	0.08	0.12	Y	+	+	+	o	Y	Y	0

Notes:

1. If the maximum fundamental frequency exceeds $0.21f_s$, it can be extended to span the full audio range by switching to a sinusoidal oscillator. For triangle and square wave oscillators a lower limit of $0.14f_s$ applies. On double rate systems, the limits become $0.105f_s$ and $0.07f_s$ respectively.

2. A comparison of synthesis methods from a complementary viewpoint is given in [14].

Appendix B: MATLAB Code

1. Bandlimited Impulse Generation using the Windowed Sinc Method

```
% parameters
fs = 48000;                             % sample rate
fc = 18300;                             % brick wall filter cut-off frequency
rlen = 10;                              % impulse length in sampling intervals
ppiv = 100;                             % points per sampling interval
beta = 9.0;                             % main window parameter
apof = 0.9;                             % apodization factor (0 = no apodization, 1 = max)
apobeta = 0.7;                          % apodization window parameter

% bandlimited impulse generation
pts = ppiv*rlen+1;                      % impulse length in points
                                        % one added to make a symmetrical impulse
x1 = 0:1:pts-1;
x2 = rlen*2*(x1 - (pts-1)/2 + 0.00001)/(pts-1);
x3 = pi*fc/fs*x2;
h = sin(x3)./x3;                        % brickwall filter impulse response
w = KAISER(pts,beta);                   % kaiser window
g = w.*h';                              % get bandlimited impulse by applying the window

% apodization and normalization
aw = 1 - apof*KAISER(pts,apobeta);
g = aw.*g;
g = g/max(g);

% diagrams
figure(1);
subplot(1,2,1);                          %*** plot bandlimited impulse ***
plot(x2/2,g);
axis([-rlen/2 rlen/2 -0.2 1.0001]);
xlabel('Time in Sampling Intervals');
title('Bandlimited Impulse');
subplot(1,2,2);                         %*** plot spectrum ***
zpad = 20;                              % zero padding factor
g2 = [g ; zeros((zpad-1)*pts,1)];       % zero pad for higher resolution
wspec = abs(fft(g2));
wspec = max(wspec/max(wspec), 0.00001);
fmax = 60000;                           % maximum displayed frequency
rng = round(rlen*zpad*fmax/fs);
xidx = 0:1:rng;
semilogy(fmax/1000*xidx/rng,wspec(1:(rng+1)));
xlabel('Frequency in kHz');
title('Amplitude Spectrum');
grid;

% markers at 20 kHz, fs-20 kHz and fs
hold;
plot([20 20], [0.00001 1], 'g');
plot([fs/1000-20 fs/1000-20], [0.00001 1], 'r');
plot([fs/1000 fs/1000], [0.00001 1], 'r');
hold off;
```

2. Compensation Filter for the High-Frequency Drop of the Bandlimited Impulse

Attach it to the end of code in appendix B1.

```
% prefilter for sample-based oscillators
figure(2);
subplot(1,2,1);
fcomp = 21000;                                  % compensate up to this frequency
rng = 1 + 2*floor(0.5*rlen*zpad*fcomp/fs - 0.5); % rng must be odd
xidx = 0:1:rng;
a = wspec(1:(rng+1));
a = 1.0./a;
ftune = 0.35;                                   % to tune out rounding errors
f = xidx/(ftune+rlen*zpad);                     % frequency relative to fs
wgt = (rng+1)/2:-1:1;                           % better fit at low frequencies
wgt =  1 + wgt.*wgt;
b = remez(16,2.0*f,a,wgt)                       % calculate 17 taps FIR filter
[h,w] = freqz(b,1,rlen*zpad,'whole');           % calculate and plot magnitude response
plot(fs*f/1000,a,0.5*fs*w/pi/1000,abs(h));
axis([0 fs/1000 0 max(abs(h))]);
xlabel('Frequency in kHz');
title('Prefilter Magnitude Response');
grid;

% check by convolving prefilter and bandlimited impulse
subplot(1,2,2);
imp = g(1:ppiv:pts);                            % sample bandlimited impulse at fs
res = conv(b,imp');                             % prefilter by convolution
res = [res zeros(1000-length(res),1)'];         % zero pad for higher resolution
wspec = abs(fft(res));
rng = round(1000*20000/fs);                     % plot overall magnitude response
xidx = 0:1:rng;
plot(20*xidx/rng,wspec(1:rng+1)/wspec(1));
xlabel('Frequency in kHz');
title('Normalized Overall Magnitude Response');
grid;
```

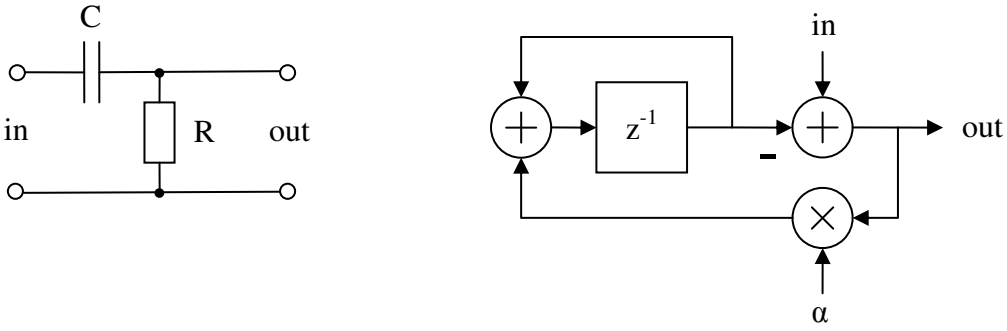
3. Bandlimited Sawtooth Segment Generation

```
% parameters
fs = 48000;                 % sample rate
fc = 15000;                 % brickwall filter cutoff frequency
rlen = 4;                   % impulse length in sampling intervals
ppiv = 2700;                % points per sampling interval
beta = 8.3;                 % main window parameter
apof = 0.5;                 % apodization factor (0 = no apodization, 1 = max)
apobeta = 0.5;              % apodization window parameter

% bandlimited impulse generation
pts = ppiv*rlen+1;          % impulse length in points = table size in words
x1 = 0:1:pts-1;
x2 = rlen*2*(x1 - (pts-1)/2 + 0.00001)/(pts-1);
x3 = pi*fc/fs*x2;
h = sin(x3)./x3;            % brickwall filter impulse response
w = KAISER(pts,beta);       % kaiser window
g = w.*h';                  % get bandlimited impulse by applying the window

% apodization
aw = 1.0 - apof*KAISER(pts,apobeta);
g = aw.*g;

% cumulative sum, normalization
g = cumsum(g);
g = 2.0*g/g(pts);
g(floor(pts/2)+1:pts) = g(floor(pts/2)+1:pts)-2.0;
g = g/max(g);               %*** desired two-sided segment ***
```

Appendix C: Miscellaneous

1. DC Trap

Audio signals may pick up a signal-dependent offset (a.k.a. bias) when they are generated or processed. A large bias reduces the dynamic range of the system and leads to unexpected, although sometimes pleasant, outcomes in nonlinear stages. However, it should be completely absent in any final output signal. Adopting the name from radio frequency engineering, we call the specific building block that eliminates the bias a DC trap. Analog and digital embodiments are depicted in Fig. 111. As the cut-off frequency f_g lies far below the sample rate f_s, we use the approximation $e^{j\omega T} - 1 \approx j\omega T$ with negligible error. Among several topologies, the one shown is preferred because the desired audio signal bypasses the filter circuitry without quantization and only a single multiplication is performed. A practical choice for f_g is about 5 Hz, which makes the trap fast enough to track offset changes but causes only a moderate attenuation of 0.26 dB and a phase shift of 14° at 20 Hz.



$$|G(f)| = \frac{1}{\sqrt{1+(f_g/f)^2}} \qquad |H(e^{j\omega T})| \approx \frac{1}{\sqrt{1+(f_g/f)^2}}$$

$$\varphi_G(f) = \arctan(f_g/f) \qquad \varphi_H(f) \approx \arctan(f_g/f) \quad ; f_g << f_s$$

$$f_g = \frac{1}{2\pi RC} \qquad f_g \approx \frac{\alpha f_s}{2\pi}$$

Fig. 111: DC Trap

2. Frequency Update

Some oscillators require a valid pair of the frequency f and its inverse 1/f at any time. Fortunately, the inverse doesn't have to be very precise as it normally only determines the bandwidth of the spectrum by defining the stretching factor of a table. If new value pairs come in at a submultiple of the sample rate, the following update schemes are recommended:

1. An independent linear fade of f and 1/f between two value pairs that are within one octave. In this case, the product is always too high during the transition (max. 12.5 % in the middle) and exact at the end.

2. An exponential fade between two value pairs: $f[n+1] = \lambda f[n]$, $f^{-1}[n+1] = \lambda^{-1} f^{-1}[n]$. The determination of λ and its inverse involves logarithms, which may turn out to be less cumbersome than it seems at first because the frequency is derived from a logarithmic mapping in synthesizers.

Appendix D: References

[1] T. Stilson, J.O. Smith, “Alias-Free Digital Synthesis of Classic Analog Waveforms”, CCRMA, http://www-ccrma.stanford.edu
[2] D.E. Knuth, “The Art of Computer Programming”, ISBN 978-0201485417
[3] R. Bristow-Johnson, “Wavetable Synthesis 101, A Fundamental Perspective”, http://www.musicdsp.org/files/Wavetable-101.pdf
[4] A. Horner, various publications on Wavetable Synthesis, http://www.cse.ust.hk/~horner
[5] E. Brandt, “Hard Sync Without Aliasing”, http://www.cs.cmu.edu/~eli/papers
[6] H. Chamberlin, “Musical Applications of Microprocessors”, ISBN 978-0810457683
[7] J. A. Moorer, “The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae”, JAES, 1976, Vol. 24, pp. 717
[8] M. Le Brun, “Digital Waveshaping Synthesis”, JAES, Apr. 1979, Vol. 27, pp. 250
[9] W. Kaegi, S. Tempelaars, “VOSIM – A New Sound Synthesis System”, JAES, Jun. 1978, Vol. 26, pp. 418
[10] G. Bennett, X. Rodet, "Synthesis of the Singing Voice", Matthews, M.V. and J.R. Pierce, eds.1989, Current Directions in Computer Music Research, MIT press, pp. 19
[11] A.H. Nuttall, “Some Windows with Very Good Sidelobe Behavior”, IEEE Transactions on Acoustics, Speech, and Signal Processing, Feb. 1981, Vol. ASSP-29, pp. 84
[12] M. Puckette, T. Apel, “Real-Time Audio Analysis Tools for Pd and MSP”, ICMC Proceedings, 1998, pp. 109
[13] J. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation", JAES, 1973, Vol. 21, pp. 526
[14] T. Tolonen, V. Välimäki, M. Karjalainen, “Evaluation of Modern Sound Synthesis Methods”, Mar. 1998, ISBN 951-2240122
[15] A. Huovilainen, V. Välimäki, “New Approaches to Digital Subtractive Synthesis”, ICMC Proceedings, 2005, pp. 399
[16] P. Schoffhauzer, “Synthesis of Bandlimited Analog Waveforms Using Frequency Modulation”
[17] O. Niemitalo, “Polynomial Interpolators for High-Quality Resampling of Oversampled Audio”, http://www.yehar.com/dsp/deip.pdf
[18] L. de Soras, “The Quest for the Perfect Resampler”, http://ldesoras.free.fr

Recommended Additional Reading and Links

1. C. Roads, “The Computer Music Tutorial”, 1996, ISBN 978-0-262-68082-0.
2. P.R. Cook, “Real Sound Synthesis for Interactive Applications”, 2002, ISBN 978-1568811680
3. J.O. Smith’s Homepage, http://ccrma.stanford.edu/~jos/
4. M. Puckette’s Book Project, http://crca.ucsd.edu/~msp/techniques/latest/book-html
5. E. Weisstein’s World of Mathematics, http://mathworld.wolfram.com
6. GSL (GNU Scientific Library), http://www.gnu.org/software/gsl
7. No Matlab? Go here: http://www.scilab.org