ELECTRONOTES 228

Newsletter of the Musical Engineering Group

1016 Hanshaw Road, Ithaca, New York 14850

Volume 24, Number 228 June 2016

WAVESHAPING – STATIC AND DYNAMIC

-by Bernie Hutchins

INTRODUCTION

Building synthesizers would be much more difficult if we had to design a different VCO for every different waveshape we traditionally use. Usually we look for sine, square, sawtooth, triangle, and pulse. Because it is useful and so easy to do (just one or two op-amps) we generally include a PWM (Pulse-Width Modulation) feature. The PWM circuitry offers, first, a manually variable (panel knob) to set an initial pulse width (or "duty cycle"). PWM is also then generally available **dynamically** (hence the term modulation). PWM is far simpler to implement than a VCF but is similar in achieving a dynamic spectrum. Recently [1] we noted that the traditional use of PWM was in an envelope-driven mode (once during a generated tone) and not as modulation by a periodic modulating waveshape (as is the case with FM – frequency modulation). This additional possibility was explored.

As we have remarked many times, static waveshapes are not only (generally) too boring to the ear but all (very roughly speaking) much the same. When we want to synthesized musical waveforms, we think of the standard waveshapes, as listed above, but we may shy away from these "textbook" shapes, having perhaps looked a while at the shapes of signals from acoustical instruments. With such investigations, we learn that these are not easy to display on a scope – they keep changing! Something **complex and dynamic** is going on [2] – no consecutive "cycles" are (in general) the same. The dynamics, in line with Moog's notion of subtractive-synthesis, was to be achieved with dynamic filtering. However, would we not be ahead of the game if we "captured" a cycles of a trumpet or violin, and used it as starting material instead of a sawtooth or square? Not to any useful degree – certainly not as much as one might expect. A few easy-to-obtain waveshapes are enough.

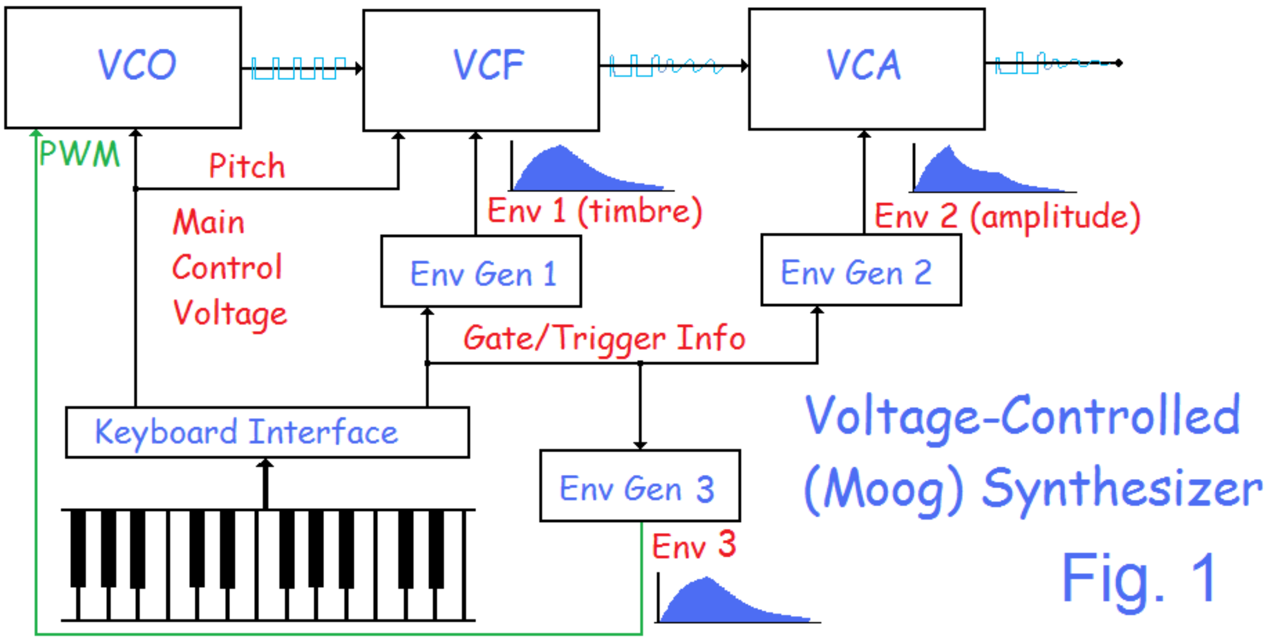


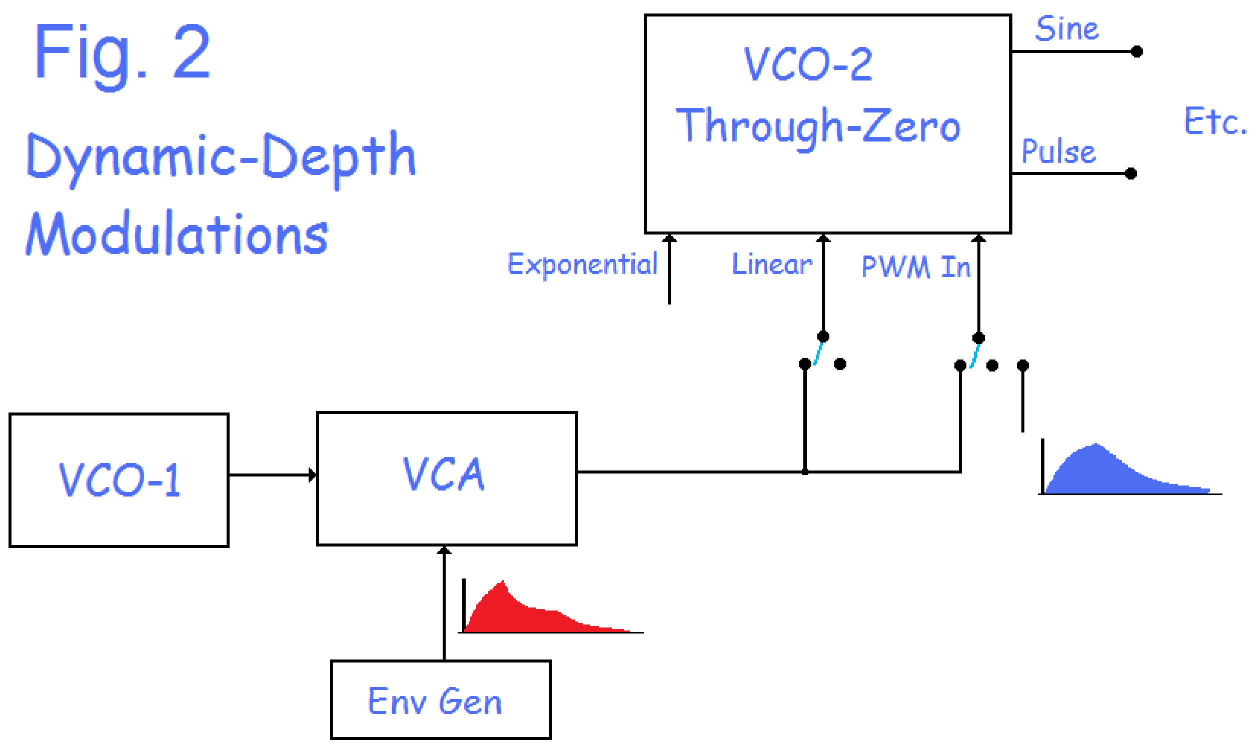
Fig. 1

Fig. 1 shows a modification of a previous figure from EN#220 [2] where we have added a third envelope generator for the PWM input of the VCO (basically a common "patch" where any envelope generator might be used in multiple controls, as a matter of economy – we just show three different ones). In the case where the PWM is not used (perhaps the output of the VCO is not the pulse, or the width is initialized and held fixed by a panel knob), the entirety of the spectral dynamics are provided by the VCF. With the green path, there is already a dynamic spectrum provided by the PWM, combining with the VCF effect.

What makes the output a musical sound (at least one that we can relate to conventional musical instruments) is the dynamics of the spectrum (PWM and/or VCF) with amplitude dynamics provided by the VCA. This is exactly the elements of "**Complexity and Dynamics**" that Moog found essential [2, 3]. Without both, the result is often (or soon becomes) too boring for the ear.

Since we would not tolerate designing separate oscillators for each waveshape desired (a half dozen perhaps) we want to start with whatever waveform we originally generate (today called a "core" oscillator which is usually triangle-core or saw-core) and design "waveshapers" to round out the ensemble. Over the years, practical methods have all been worked out, and these we will review below. We think of these as static (hard wired). As we said, PWM is variable and often dynamic, but classed with the standard waveforms, at least in terms of a panel layout.

Fig. 1 relates to the "subtractive synthesis" model that is classic. Immediately with the introduction of the voltage-controlled concept, was the notion that frequency modulation



(FM) and other types of modulation were possible as an alternative method of spectral dynamics (or in addition to filtering). At first, dedicated to the generation of "clangorous" sounds (like bells) the general FM notions proliferated. FM methods were capable of producing complex spectra, but particularly with regard to the dynamic cases (particularly as with dynamic depth through zero) some astounding new sounds became possible [4-6].

Fig. 2, also from [2], shows the familiar idea of through-zero dynamic depth FM and relates it to PWM as it is traditionally used with an envelope. Note that both move the spectral dynamics generation tasks to the VCO (or are in addition to any VCF).

So what is a waveshaper? Certainly it is any of the numerous circuits used to "convert" one static waveform to another (such as triangle to saw). In an extended sense, something like an ordinary filter shapes a waveform (except for a sine – no linear filter changes this "eigenfunction"). However, in general a fixed filter will not give a scaled waveform for different input frequencies. The notion of a filter (specifically a VCF) as a waveshaper is thus suggested as restricted to a VCF tracking a VCO (Fig. 1), and is usually dynamic within the tone. (Using a VCF as a waveshaper for a VCO would not be efficient!) At the very outside, I guess any modulation could be considered a form of waveshaping.

Somewhere in the middle are a number of devices that look like descendents of ordinary fixed waveshapers with added parameters that were usually voltage-controlled. These were often called "timbre modulators" and many such ideas were included in the Electronotes pages [7].

The static waveshapers were "single channels" in that they had a single input and a single output. Some of the circuits (or which PWM is the prime example), had variability, at least to a controlling panel knob, and more usefully, as a voltage-controlled option.

One class related to waveshaping was comprised of the various "animator" devices [8]. These were designed individually, and with a largely empirical investigative methodology. The idea was, generally considered, to emulate the "fat sound" that could be achieved by putting a good number or VCOs in parallel, nominally tracking, but inevitably, and to advantage, having small tracking errors. (So they "beat" slowly, somewhat as one might expect the various violinists in an orchestra's string section to converge about a varying consensus target pitch.) Such devices, as analog circuitry, were cumbersome, but the Multi Phase Waveform Animator (MPWA) [9] was quite popular. In contrast to single channel devices, the animators had parallel sections, in imitation of multiple sources of similar but non-identical content. The MPWA used 8 channels. It was felt that just 2 or 3 channels were not enough. Using eight channels was quite nice. Sixteen channels (two MPWAs in parallel) was too much – too homogeneous.

Ian Fritz (long-time contributor to our art and most likely the ranking active expert on analog synthesis) recently emailed me, in connection with a phase shifting as waveshaping patch that he had tried, the MPWA sawtooth shifter using just two parallel channels. Unlike my 8-channel MPWA Ian had not hard-wired the shifters to fixed LFO controls. This meant that he could patch in external controls, <u>including audio-frequency signals</u>. His email subject was "one we missed?" and most interestingly, he said this gave him a result he had never heard before. <u>That's unusual</u>.

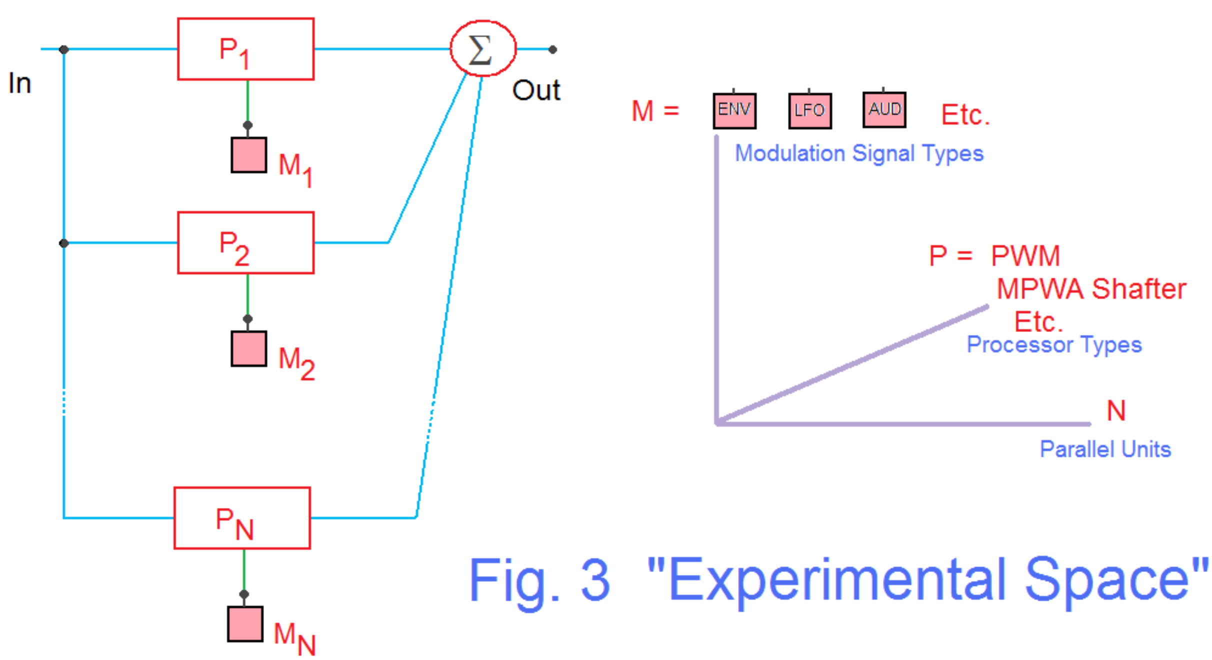


Fig. 3 "Experimental Space"

In addition to something specific to consider, Ian's comment caused me to consider that there was possibly a broad area of "experimental space" to explore. The space was in the two abstract dimensions of "Processor Types" (P) and "Modulation Signal Types" (M) as well as an ordinary third dimension of number of parallel units (N). See Fig. 3. For example, a well-explored regions would be N=1 and M=ENV with P=PWM. Also explored would be N=8, M=LFO, with P=MPWA. Recently investigated [1] would be N=1, P=PWM, with M=AUD. Lots of unexplored space.

REVIEW OF STATIC WAVESHAPERS

We start off now by reviewing the traditional waveshaping ideas. Here we have in mind that we start with a VCO that has either (1) A sawtooth "core" or (2) a triangle-square "core". The oscillator that is inherently a sawtooth is a linear ramp with a reset when the output reaches a set level. The triangle-square oscillator is the classic loop of an integrator and a Schmitt-trigger, where the Schmitt-trigger reverses direction of the integrator's ramp (hence a triangle), in which case both a triangle and a square are available directly. If the three waveforms mentioned here, only the sawtooth and the triangle are useful for additional waveshaping. The square is useful only because we have an accompanying triangle. Why is the square not useful?

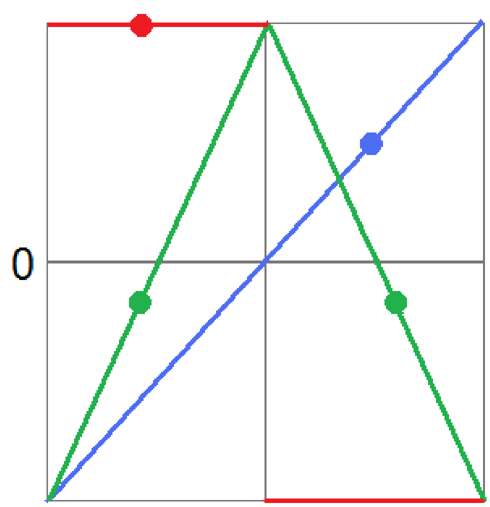


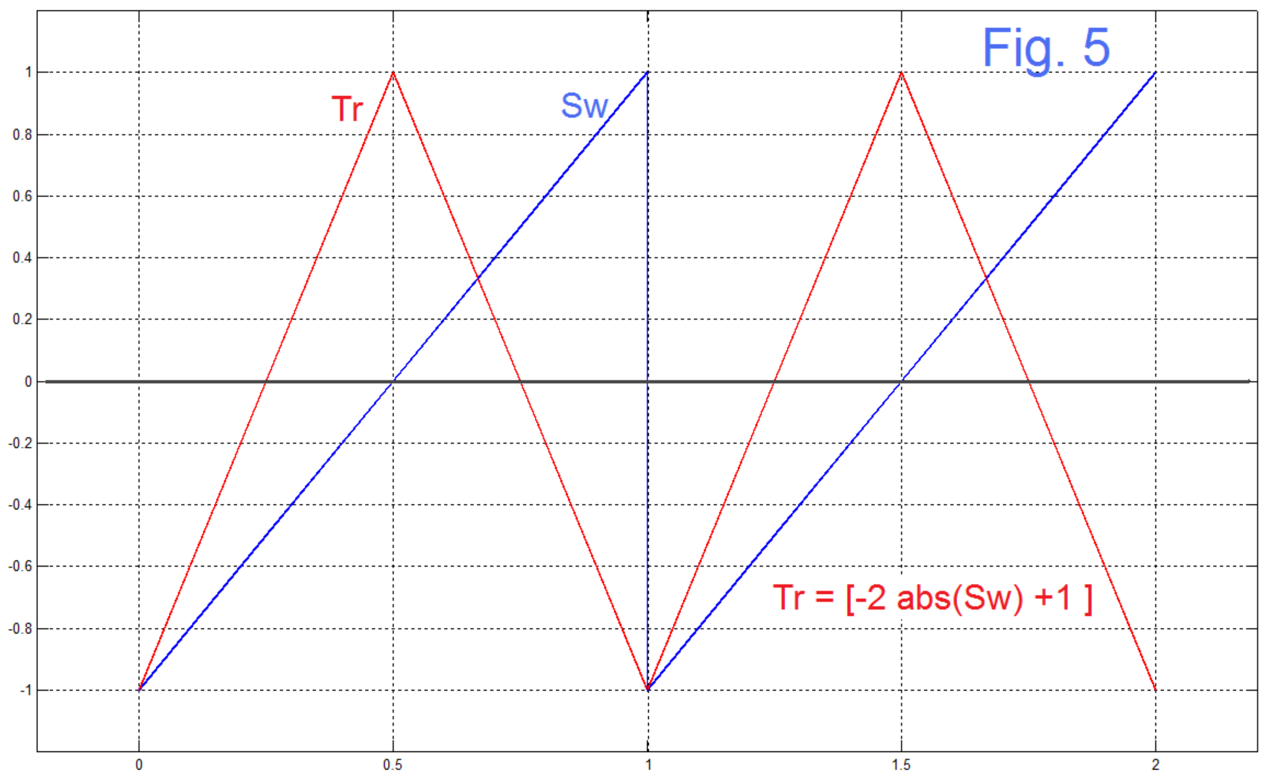
Fig. 4

The "information" in knowing where you are in a cycle

Fig. 4 shows a single cycle of square (red), a triangle (green) and a saw (blue). Suppose we know an instantaneous value of some point within the cycle of these. Clearly knowing the value of the sawtooth (blue dot) tells us exactly the current phase. Knowing the value of the square (red dot) tells us only which half of the waveform we are in (top as shown). The value of the triangle is an ambiguous pair of green dots. This makes the sawtooth very useful, and the square worthless, when it comes to <u>additional</u> waveshapes. What saves the triangle here is that if we know <u>both</u> the triangle and the square, the phase is determined. But you will shout, the square must have exactly the right phase like you show. Exactly. And it is because we are talking about a Triangle/Schmitt-trigger, where the integrator is driven by the square (as in Fig. 4) that we have, automatically, both waveshapes <u>and</u> the desired phase relationship. This leaves us with the tasks of converting triangle-to-saw and vice versa, as necessary, or obtaining a pulse (as PWM) and a special case square wave (if necessary), and of shaping a triangle to a sine.

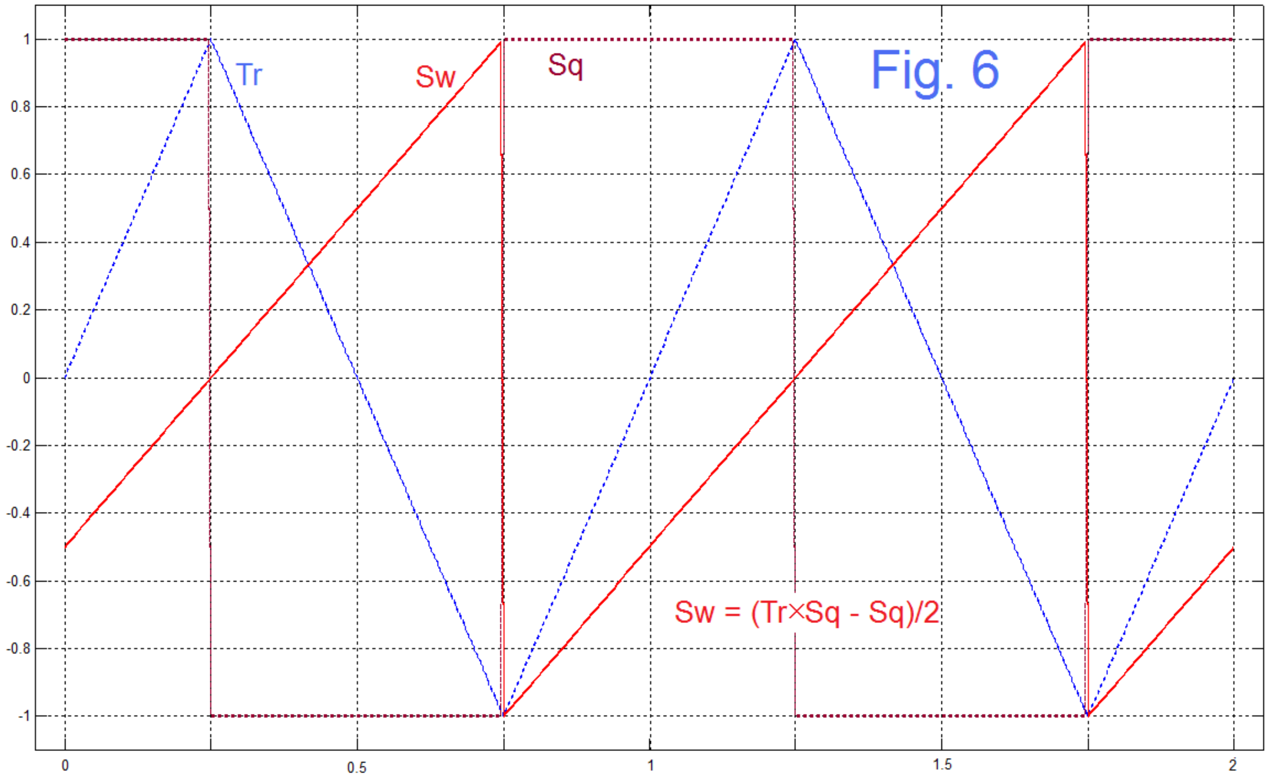
The tasks of designing waveshapers can be divided into two steps. <u>First</u>, we have to make it work on paper, and this usually means quite literally to sketch out the situation on grid paper. Alternatively, we want a math expression: like we want to add/subtract, scale, shift, invert, or take absolute values. The <u>second</u> step is to assure ourselves that we have the necessary circuitry. We can certainly do arithmetic with op-amps. In addition, precision full-wave rectifiers with op-amps do absolute value. A voltage-switchable inverter/non-inverter is available with a FET switch. And of course, an op-amp is a perfectly good comparator when used open-loop. Just the right tools.

The easiest of the converters is the sawtooth-to-triangle. Looking at Fig. 4, we immediately see that if we take the absolute value of the sawtooth (blue) it is already a triangle. We just need to adjust the gain, sign, and DC level. (A simple one-transistor circuit to do this shaper was in fact invented by Moog [11]). Fig. 5 shows two full cycles of the converter, and an example of actual circuitry is found in [12]).



The result of Fig. 5 is clear enough, and it might seem that we just invert the equation there to obtain the saw from a triangle. It's not that simple. First we can't invert the abs value without a knowledge of the sign. Secondly, as we shall see in a moment the phase relationship between the corresponding saw and triangle are not the same in both cases.

Fig. 6 shows the triangle-to-sawtooth shaping scheme, and note that the square wave is also involved. This square wave we happen to have already as part of the oscillator. [If it were not, we could entertain getting it as the sign of the derivative of the triangle.] It is useful to also recognize that an intermediate step is a “double frequency” saw. Because



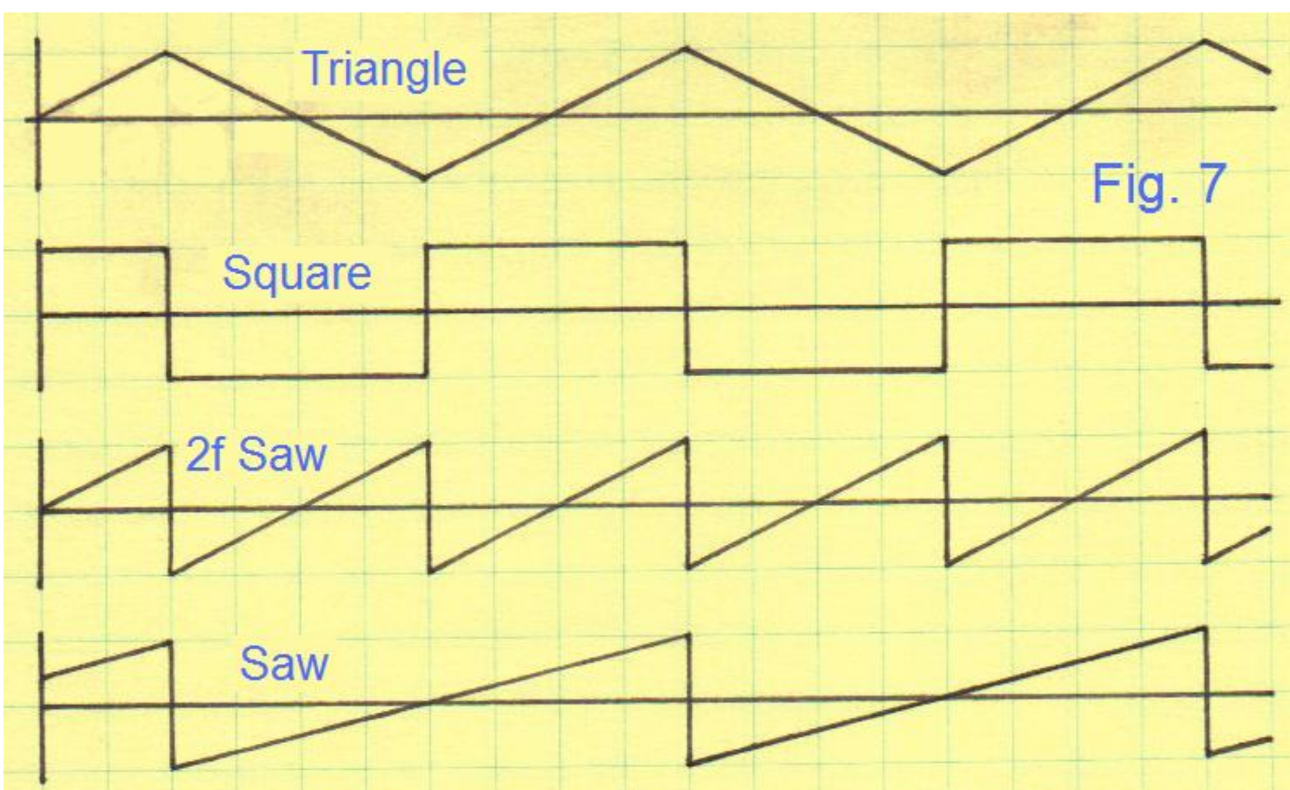


Fig. 6 is already cluttered enough, we have borrowed a more detailed, four-waveform figure from EN#75 [12]. At the same time this older diagram, Fig. 7, makes it clear that the saw and the triangle have a different phase relationship in this shaper relative to Fig. 5. In most cases, we don't worry about the phase, as the "timbre" (tone color) of the waveform is independent of phase. It is even largely independent of the relative phases of the harmonic components – the so-called "Ohm's Acoustic Law" or the phase deftness of the ear.

Examples of the actual circuitry for these shapers used in Fig. 5 and Fig. 6 are found in VCO circuits [12]. As we suggested, we need to first try a mathematical expression for the waveshaper we want, and then see if we have a circuit that does this. Turning now to the task of shaping a triangle into a sine wave, we note that we would just use:

$$Si = \sin(\pi Tr)$$

which is mathematically simple enough. The triangle just becomes the phase of the sine. However we have no such conversion circuitry (at least not simply). It is true that a low-pass filter could convert a triangle (or other waveshape) to sine, but to do this properly, we would require a tracking VCF for the filter. Instead, we resort to any of a number of "tricks" to bend the triangle tops around a bit at higher amplitudes. A FET is sometimes used for this [12] although a differential amplifier input stage (using a CA3080) is also popular [10]. various other choices for the nonlinear transfer curve are known [13].

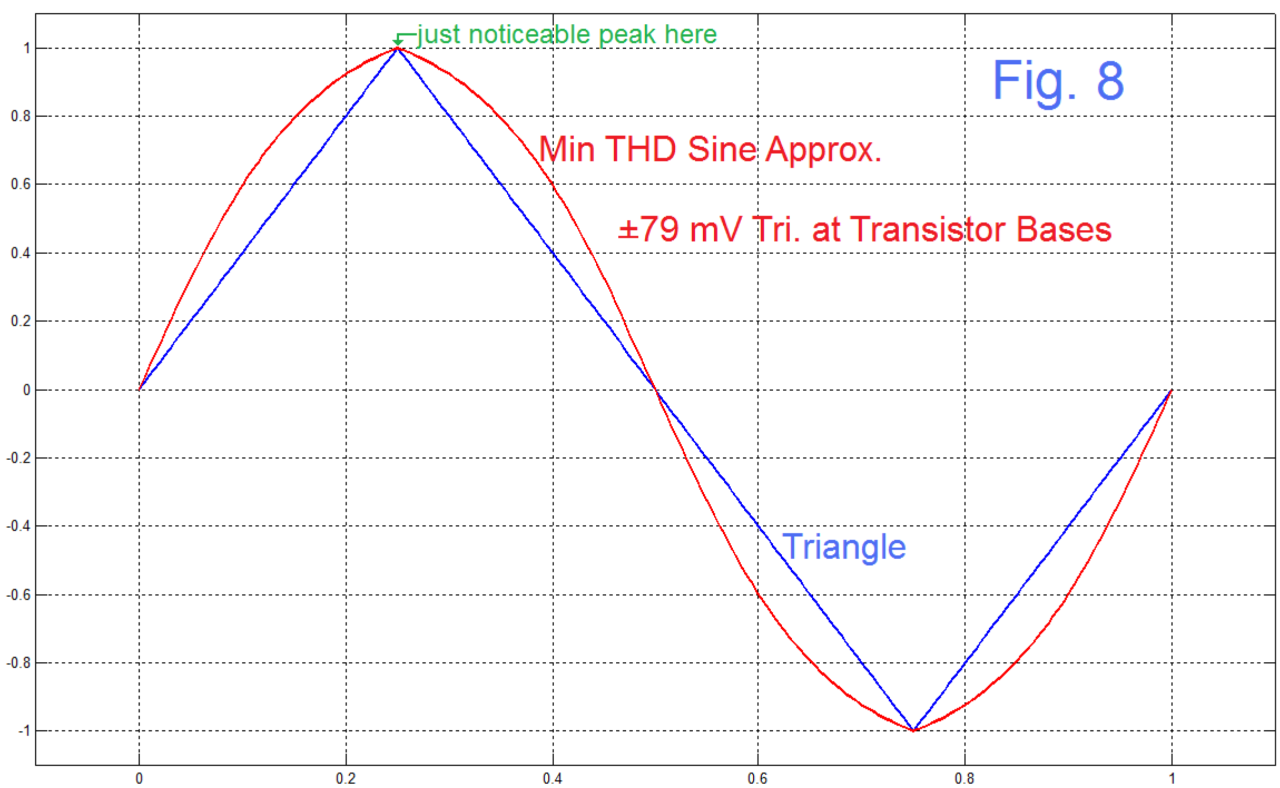


Fig. 8 shows the use of the differential amplifier with an optimized peak-to-peak input for the triangle to 79 mV. The output voltage is given as:

$$v_{out} = (2/A)\left(-0.5 \ + \frac{1}{1+exp\left(-v_{in}*\frac{0.079}{0.026}\right)}\right)$$

where:

$$A = 2*\left(-0.5 \ + \frac{1}{1+exp\left(-\frac{0.079}{0.026}\right)}\right)$$

That's about it, except for the pulse and square wave.

RECTANGULAR SHAPES

Unlike the triangle, sawtooth, and sine, the waveshapers for pulse and square (a special case of pulse) are different in that they destroy a good deal of information (Fig. 4). The square wave is simply obtained with an op-amp used open loop as a comparator, and it can be driven by any of the waveshapes (typically triangle).

What is really different here is the pulse waveform <u>because it has a parameter</u> – a reference level which determines the width from 0% to 100%. For the most part, the various widths are considered to be of high harmonic content. So any VCO with a pulse option would have (at least) a knob to set a width and one can experiment with the resulting tone colors. If the knob is set and not changed, we have just another waveshape choice added.

Fig. 10a indicates the basic comparator. For a square output, the reference would be set to 0 (just grounded omitting the summer). This square would be a 50% "duty cycle" pulse. In the case of the pulse, the reference level is made variable between the limits of the driving waveshape (for 0% to 100%) and is a "parameter" (a voltage that controls) the pulse width. Because the reference is now considered a voltage (from many possible sources), the process of controlling the width is called a modulation or PWM. This distinguishes the pulse from the triangle, saw, sine, or square. A whole new world. Because of the extensive variation of tone color possible, and the simplicity of the circuitry involved, this is popular. It is possible to obtain a dynamic spectrum approaching that of a subtractive synthesis mode with a VCF. But, as mentioned, the traditional use has been limited.

To be more clear, looking back to Fig. 3 here we have used principally N=1 and M=ENV. The dynamics is achieved <u>once per musical tone</u>. We did get around eventually

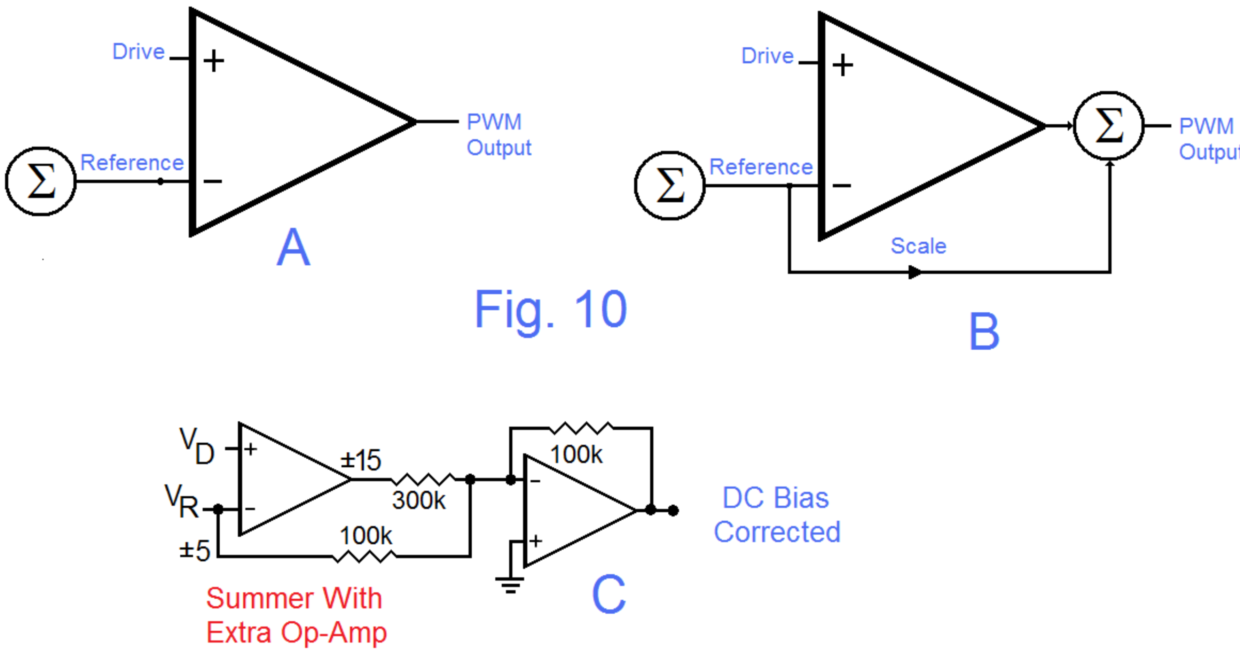


Fig. 10

to considering PWM with M=AUD in EN#216 [1] and that proved of some interest. One thing that came up was the fact that the conventional PWM has a DC term. This was no surprise. A 0% pulse would be a maximum negative bias, a 50% (square wave) would be zero bias, and 100% would be a maximum positive bias. The bias level is the same as the reference level. It is probably clear enough why this is, and is demonstrated in the reference [1]. Because the component is DC, or perhaps has only the very low frequencies of an envelope, the result was inaudible. Thus with N=1 and M=ENV we always ignored it. It is however, easy enough to remove the bias [1]. This DC bias might well be a problem if the PWM output becomes a control to be used further on. { A related issue is that we have ignored the choice of drive even though it had an small tilting effect on the spectrum [1]. It did not matter because there was only the very slow envelope-controlled variation. }

Fig. 10b shows how the reference can be added to the output to remove the DC bias. Fig. 10c was a suggested circuit. Note that the op-amp output is usually less than ±15 (perhaps ±13.7 on 15 volt supplies) so the 300k resistor would be smaller (perhaps 270k) in a real circuit.

Fig. 11 shows some additional details of the removal of the DC bias. Here there are four plots for four different example reference levels (red) all for the same triangular drive (green). In gray, we show the resulting ordinary PWM. Note that the triangular drive and the ordinary PWM always take up the range, and only that range, between +1 and -1 for our normalized examples. Now, the blue waveform is just the gray waveform offset by the reference. Note that it does range outside the ±1 range of the drive and ordinary PWM.

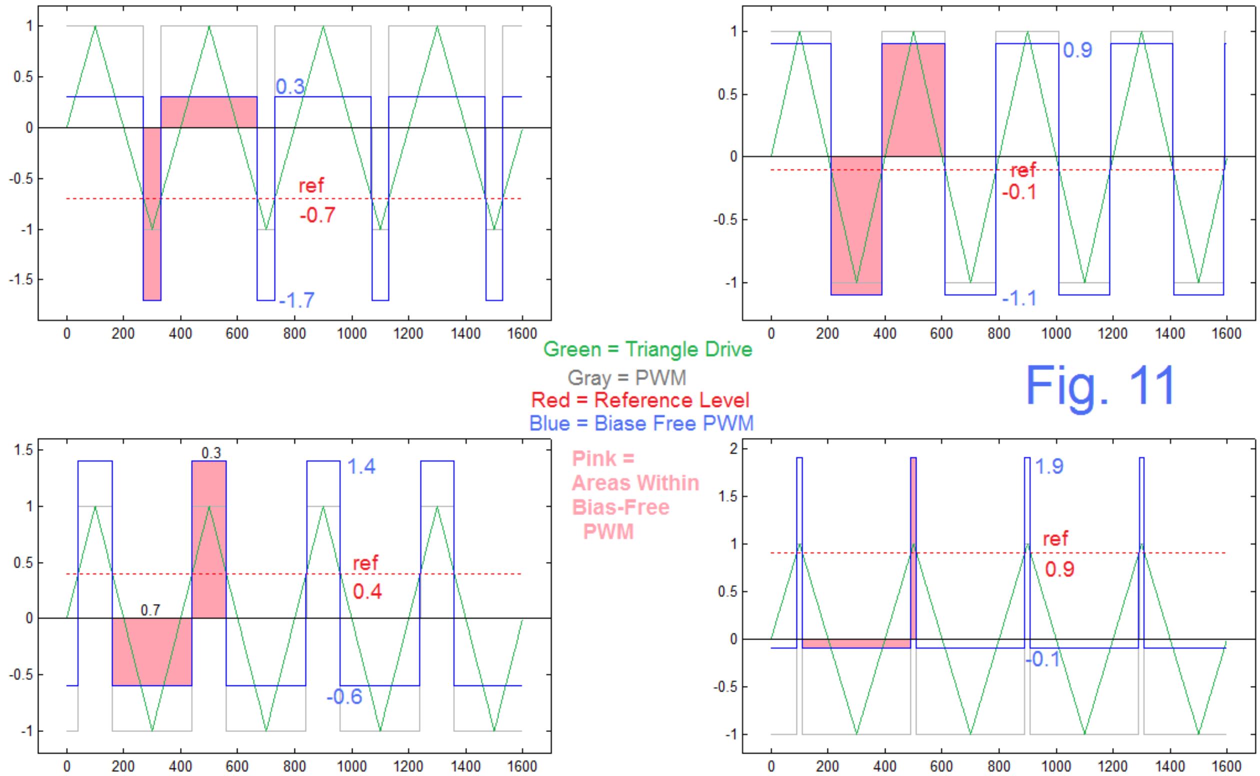


Fig. 11

The extremes of the pulse would have to be different if different widths were to balance to 0 DC. In the examples, note that one cycle of the four cycles shown for each example has its areas shaded in pink. The area above and the area below are the same. For example, for the 0.4 reference, the output (blue) is below the zero line (black) for 0.7 of the cycle and has value -0.6, and is above zero for 0.3 of the cycle, with value 1.4. The total area is thus $0.7 \times -0.6 \;\; + \;\; 0.3 \times 1.4 = 0.$ No net DC. And so on.

TIMBER MODULATOR EXAMPLE 1 – Ian Frtiz's Double Pulser

At this point we have we have updated the (decades old) traditions of waveshaping along with some more recent examinations of classic devices. We want to now continue by looking at some "timbre modulator" ideas we presented but did not analyze in detail originally (if for no other reason, we lacked good tools).

The first of the Timber Modulators is the Double Pulse Shaper as submitted by Ian Fritz back in EN#73 [7a]. What Ian (Fig. 12) did was to form two PWM shapers which operated on both polarities of the driving waveform (a sine as shown but also a triangle is suggested as basically the same). Thus one pulse is generated about the positive peaks of the drive and a second pulse (in the opposite polarity) is generated about negative peaks of the drive – two pulses in the same cycle.

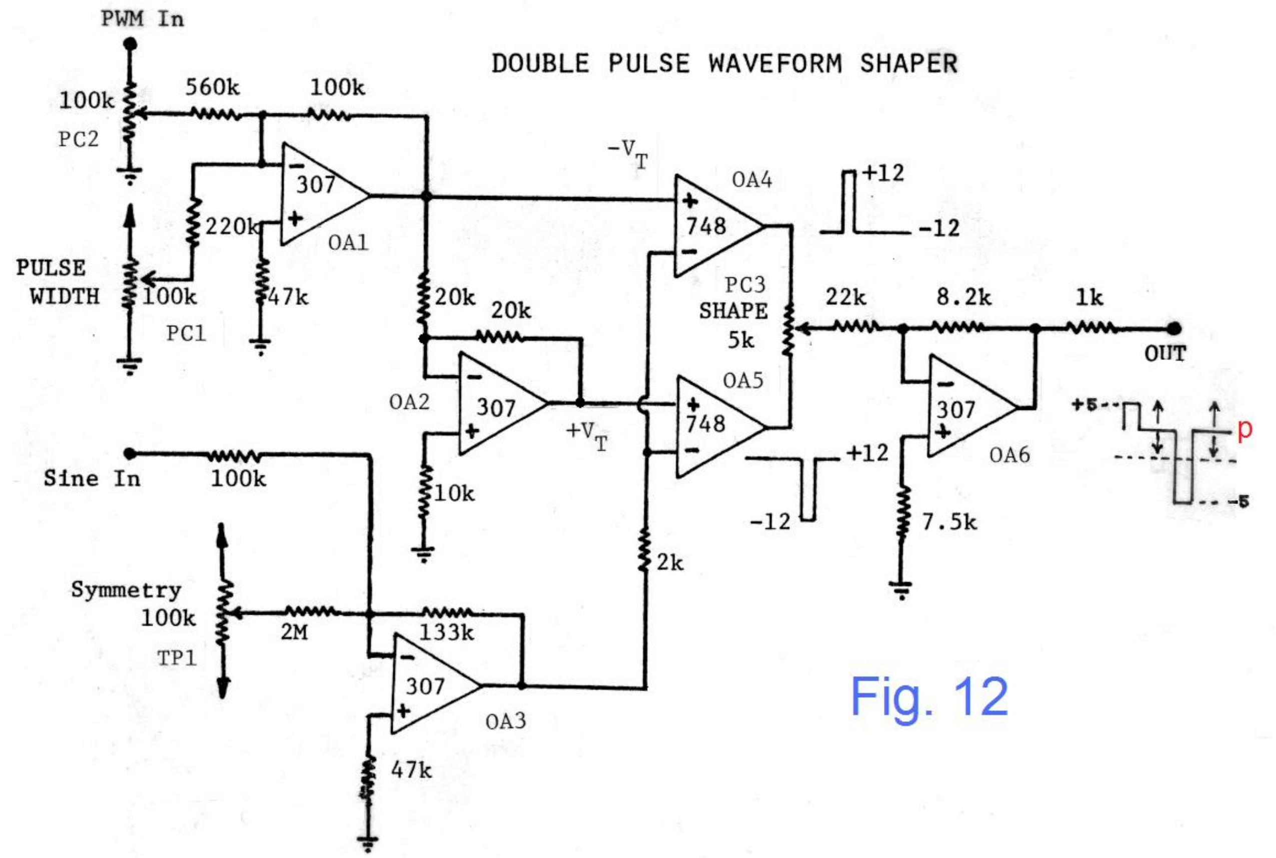
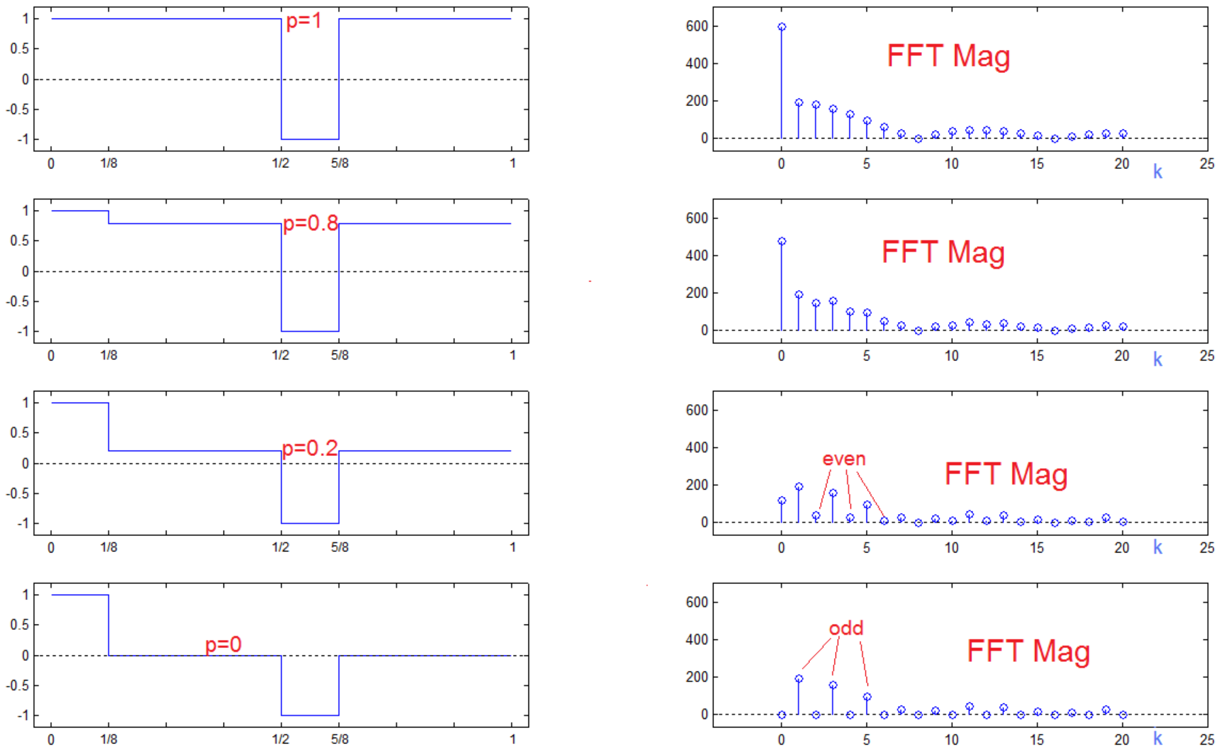


Fig. 12

So, in the circuit of Fig. 12 the leftmost five op-amps comprise the complementary set of PWM circuits while OA6 allows a cross-mix of the two. In a normalized view, the output can be considered a unit length cycle, with pulse extremes (of a parameterized width),at +1 and at -1, with a parameterized "plateau" level (p) set by the mix. This cycle of course has a Fourier Series (FS) representation which would not be difficult to derive. The result would be a formula that would, however, be difficult to interpret just by looking at it. Accordingly, we would need to compute and plot some result, as a function of the parameters. It is easier to approximate the FS employing a discrete version and using the FFT.

First we can note that we know certain limits. When we have the "Shape" control at the top or the bottom, we have ordinary PWM with a spectrum that contains all harmonics. Potentially, if the "duty cycle" (width divided by cycle length) is a rational fraction (as it will be in a discrete approximation), a few harmonics are missing. For example, if the duty cycle is 1/8, the 8^{th}, 16^{th}, 24^{th},… harmonics are nulled out. For a general setting of the "Shape" control we have a mix of two pulses, with a DC level. Due to linear superposition, we have no new harmonics in the mix. It is found that all the harmonics will have changed amplitudes (and phases), and some may cancel. We can see that this does indeed occur just by noting if the duty cycle is 1/2, and p=0, we have a square wave that is missing all even harmonics. Thus by cross mixing, a variable spectrum, in addition to the PWM, is achieved. This plateau level can be electronically cross mixed (the pot replaced with complementary VCAs). Indeed, Ian reports doing this and achieving an interesting dynamic spectrum as a result.



Timbre Modulation Due to Plateau Parameter

Fig. 13

The PWM function is well-studied, so here we look at the effect of the mix (of the plateau in the more direct construction). Fig. 13 shows the case where we have a duty cycle of 1/8, and plateau values of +1, +0.8, +0.2, and 0. That is, we take snapshots of a modulation from p=1 to p=0. In the top left panel, we have just the 1/8 duty cycle pulse, and the spectrum (FFT Magnitude) on the top right shows as expected, a sync-like roll-off with harmonics 8 and 16 missing. When we make p=0.8, a small change, we see a small change of the spectrum, but note that we start to see the decline of the even harmonics (including k=0, DC). More dramatically, at p=0.2 the even harmonics are becoming very small, and at p=0, they are all gone. So the result at p=0 is all odd harmonics. These are the same harmonics you get with a square wave, although the amplitudes are different.

It is quite interesting that even harmonics drop out (although evident from the time symmetry). Our interest in timbre modulators is, after all, an evolving spectrum rather than static waveshaping.

ASIDE No. 1: ON EVEN ONLY HARMONICS

It's perhaps an "old joke" to discuss a waveform that has only even harmonics. A waveform with all harmonics would have a fundamental 1f with harmonics 2f, 3f, 4f, 5f.… . We hear this as a pitch of 1f. It is further true that the pitch is much more easily perceived when not just the fundamental is present (a sine wave of 1f) but where this is supported by numerous harmonics like a sawtooth or pulse). It is also true that the pitch of 1f is strongly supported by a waveshape such as a square wave which has odd harmonics (indeed also the p=0 case of Fig. 13). Such odd-only cases are said to be “woody” or “hollow” perhaps as they are remindful of a clarinet sound.

What about a waveform that has only even harmonics such as 2f, 4f, 6f, 8f….? Enter the joke that this is really all harmonics of a fundamental of 2f. Fig. 14 shows a comparison of odd harmonics falling off as 1/k, k being the harmonic number, which is a square wave, each showing two full cycles of the reconstruction. This is classic FS, and produces a pitch f. If we then form the series using only the even harmonics, (again falling as 1/k), we get the sawtooth seen in the lower panel (blue) of Fig. 14. This shows four cycles of a sawtooth. In isolation, this produces a pitch of 2f, as would seem necessary. That is, it has all harmonics of 2f. Hence, as a static waveshaper, little or nothing is added. That’s the “joke”. Thus we instead emphasize that we are interested not in the static case but rather in (at least) a spectral animation that makes a “benchmark” aural impression equivalent to a VCF in flat low-pass mode.

To show exactly what we did here, we show the Matlab code in colors corresponding to the colors in Fig. 14. To make the final point, we added the fundamental to the sawtooth sum (red).

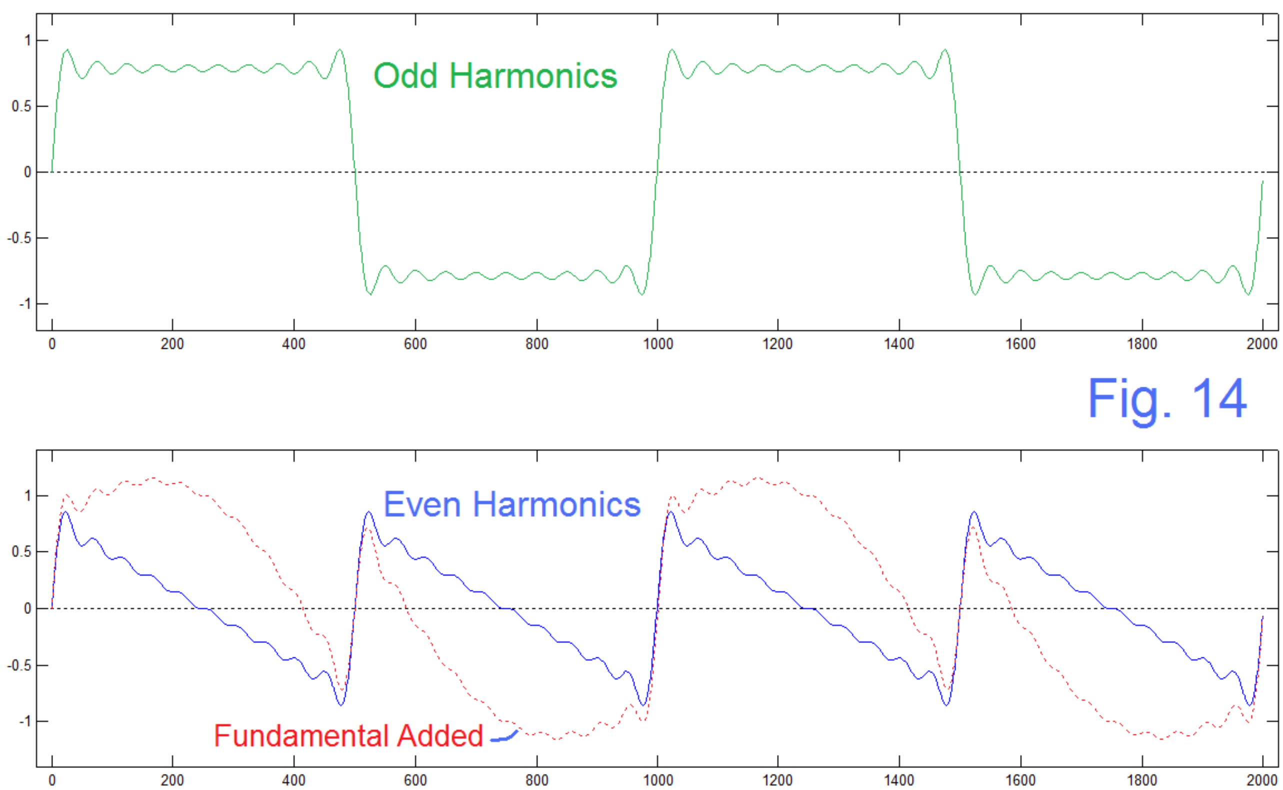


Fig. 14

```
for k=1:2:19
  xo=xo + (1/k)*sin(2*pi*k*[0:1999]/1000);
  xe=xe +(1/(k+1))*sin(2*pi*(k+1)*[0:1999]/1000);
end
xe1=xe + sin(2*pi*(1)*[0:1999]/1000);
```

TIMBRES IN TRANSITION

The point is now that our perception of pitch, or indeed of the timbre (tone color) of a sound with dynamics has a history involving the impression the sound has been making for perhaps 100 ms to 1 second. So even if the double frequency sawtooth would be heard as a pitch 2f in isolation, as the spectrum progresses through transitioning situations involving all harmonics (pitch 1f), this pitch is "etched" into our ear/brain to the point where it is not disrupted to 2f by a short (perhaps instantaneous) full cancellation of the fundamental. As a better known example, recall that with dynamic depth FM, a fundamental (or any other frequency perhaps) can disappear and reappear without disruption of a smooth impression.

ASIDE No. 2: SIMULATION OR CONSTRUCTION

Very likely most readers here are familiar with the issues of discrete-time sampling and accept the notion that the information in a signal <u>can</u> be represented by time-samples. (Today, few if any of us are listening to music played from an analog source.) A parallel to this in a visual sense is that if you look at graphs of signals of the type I have included above, you suspect that I did not draw these with pencils. In many instances today we have a plot of discrete samples (like dots, "stems", "lollypops", or stars), intentionally indicating the discrete nature, while at others we want to represent (have the computer "draw") continuous signals. In this latter case, we calculate a dense enough set of samples that the plot looks continuous on the screen. For example, the left sides of Fig. 13 are represented by 800 samples (even then by some pixel-by-pixel editing). This left side is intentionally shown as continuous, while the right side is discrete in frequency, a FS. Fig. 14 has 2000 samples left to right as indicated, "drawing" a continuous waveform as a dense set of points. We probably don't even think about this a second time.

There is a second reason (other than fooling the viewer of a plot) for representing a signal by a likely highly oversampled (relative to the sampling theorem) set of samples: we want to approximate a spectrum reasonably well (like a FS). Given that the left sides of Fig. 13 are 800 time points long, the corresponding FFTs are also 800 points (401 unique frequencies). The amplitudes fall off very rapidly with increasing k and we need only show about 25. It is also the fortuitous case that this is more than enough so that the FFT is an excellent representation to the FS [14], the FS being our preferred means of displaying the spectrum of a periodic waveform.

One additional point is that representing something <u>like</u> the left side of Fig. 13 could be simplified with 16 rather than 800 samples as: [1 p p p p p p p -1 p p p p p p p]. Here we have divided a single cycle into 16 samples. If we have in mind that each cycle need be represented by just over 2 samples, we misunderstand the sampling theorem. This notion of 2+ samples applies to a sine wave cycle. Indeed, 16 samples support a frequency that is bandlimited to less than 8. So a sinewave of frequency 1 would be okay. But a pulse is not, and can never be supported since the sharp transitions have infinite bandwidth. We are out of luck. You can't do pulses, or any waveform with a discontinuity (no sawtooth or square) or even a waveform with a discontinuous derivative (no triangle), or even one of finite length. So digital audio is useless – so it seems.

Of course digital sampling is useful – indeed essential. We know that it is just necessary to first bandlimit (<u>for practical purposes</u>) the signal to be sampled with an "anti-aliasing" low-pass, discarding some high frequency information and doing without. This we justify, for example, by noting that there is an upper limit to what we can hear in an audio application, and similar realistic limits in other sampling situations. Basically it is a matter of recognizing, intuitively and correctly, that taking a LOT of samples is sufficient. Note as well that the corresponding recovery from samples is the job of a "reconstruction filter" or a "smoothing filter", sometimes erroneously and confusingly called an anti-aliasing filter – exactly wrong. So, what could the samples [1 p p p p p p p -1 p p p p p p p] mean?

There are two ways to consider this. <u>First</u>, what continuous bandlimited waveform when sampled goes through the points [1 p p p p p p p -1 p p p p p p p]? <u>Secondly</u>, what happens if we pass the sequence [1 p p p p p p p -1 p p p p p p p] through an appropriate bandlimiting low-pass filter? Both give the same answer, at least when we take "bandlimiting" in the sense of a DFT (FFT). The answers here result from interpolation of the time points.

Here we use FFT interpolation [15, 16] where the number of time points is increased by zero-padding the exact <u>middle of the FFT</u>. In our examples (Fig. 15 and Fig. 16) we have length-16 <u>time</u> sequences (lollypops) and place, as an example, 240 zeros in the FFT middle so that when we take the inverse FFT we obtain 240 additional time points, 15 additional points between each original pair (red continuous-appearing lines). This is the answer. We consider this FFT interpolation procedure as a low-pass filtering. This does not mean, of course, that the samples came from a sampling of the red waveforms. Indeed, we just typed in samples, having in the back of our minds something like the left side of Fig. 13. Considered as continuous waveshapes, these waveforms as on the left side of Fig. 13 are rectangular and can could not be bandlimited (hence, technically, they can't be sampled). The red waveforms in Fig. 15 and Fig. 16 (for p=0.3 and p=0) are the only possible shapes that are bandlimited, in the sense of the DFT, that go through the 16 original time points. Perhaps we should mention that other interpolation methods could be used [17] with fairly similar results, and that, <u>ultimately</u>, it is the reconstruction low-pass that matters.

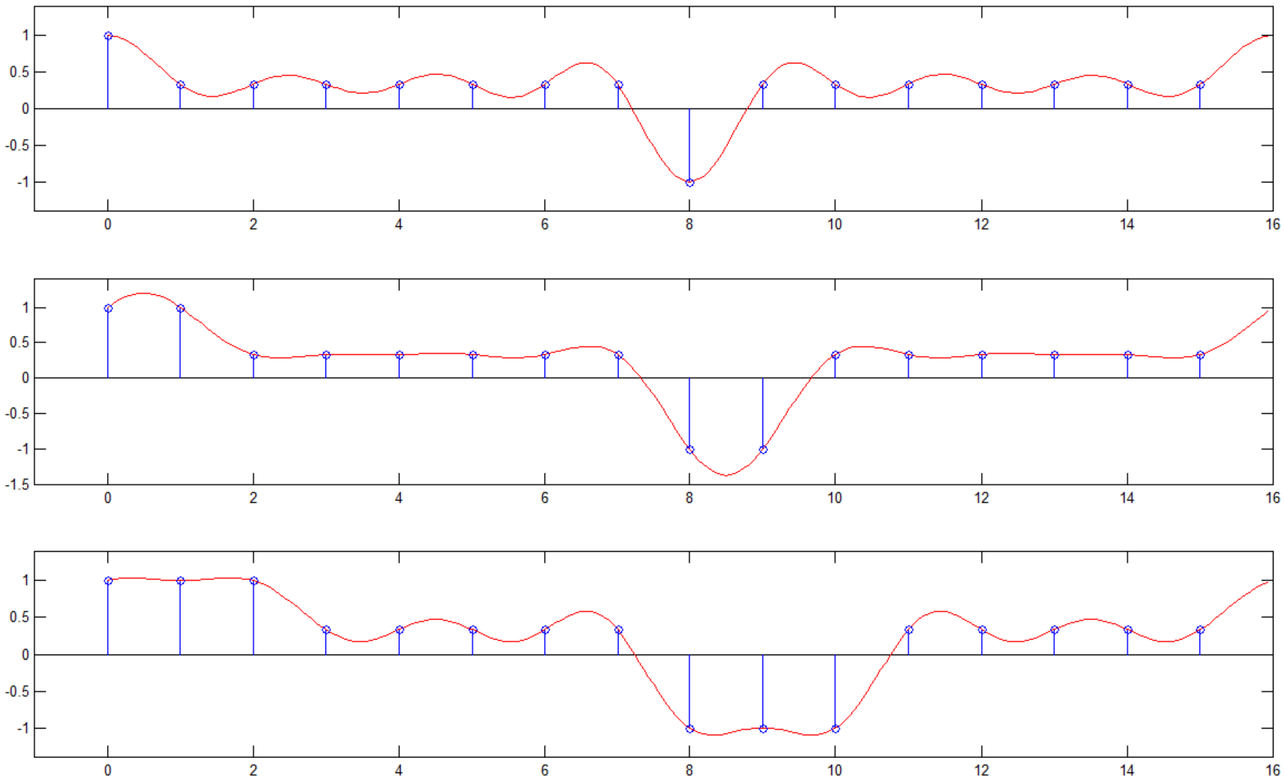


Fig. 15

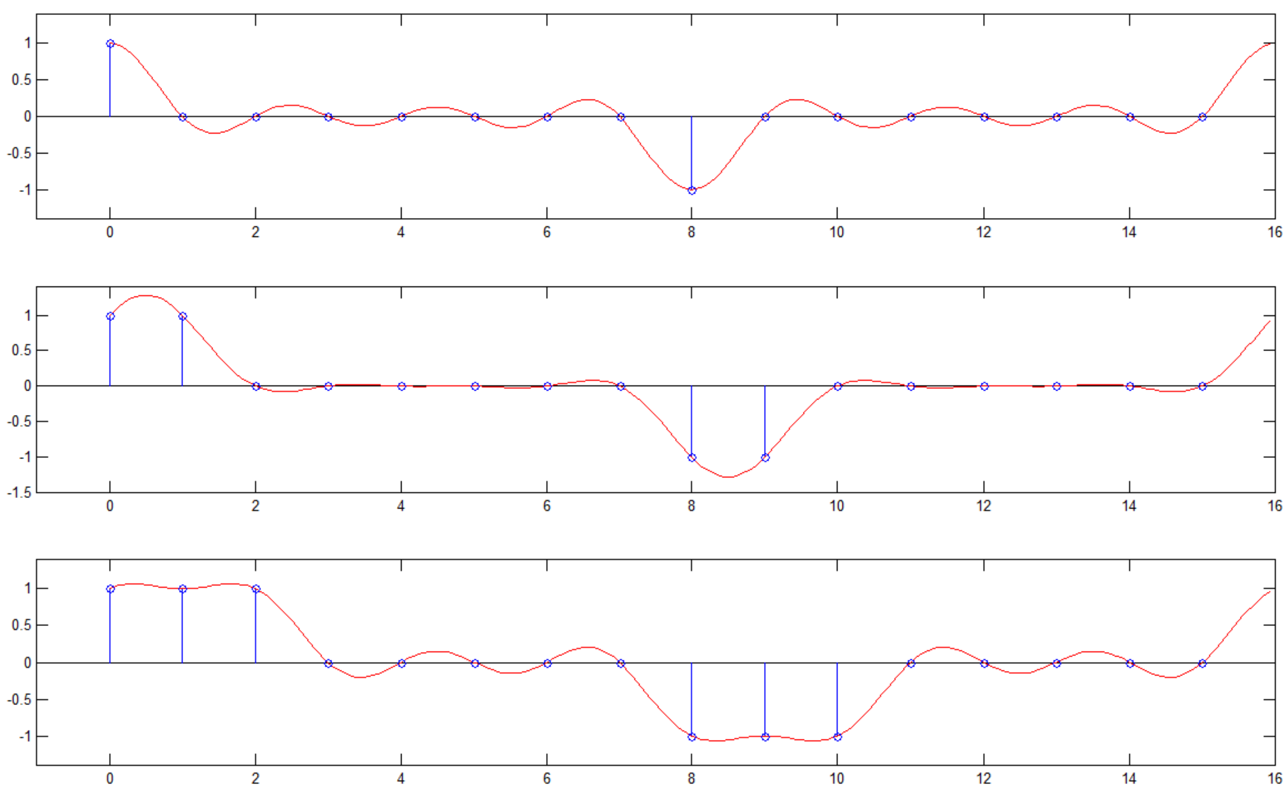


Fig. 16

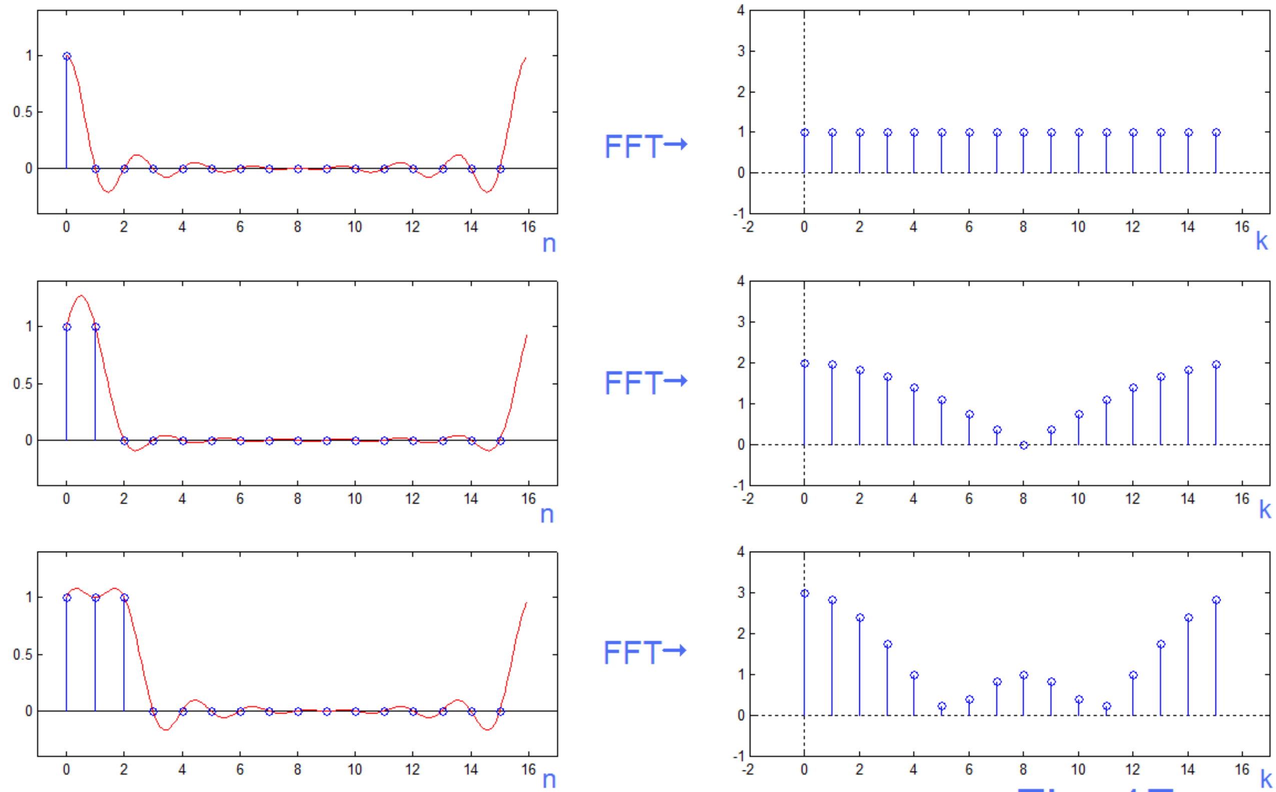


Fig. 17

The results of Fig. 15 and Fig. 16 are mainly to remind us of the bandlimited nature of the discrete sequence once they are reconstructed. In Fig. 17 we go from the double pulse shaper back to ordinary PWM and show the interpolation and FFT of three cases. If we are actually thinking of the left sides of Fig. 17 as samples of a pulse, it is clear that the actual pulse width is ambiguous to the width of one sample, and that accordingly, the spectra as shown by the FFTs n the right side are "snapshots" of the typical sinc-like spectral envelopes we are very familiar with.

SAMPLE RUNS AND LISTENING TESTS

At this point, we have looked at the double pulse shaper and know the general resulting spectrum as shown in Fig. 13. We are in a position to follow-up either by making more plots, by listening, or both. Bearing in mind the cautions of Figures 15-17, we propose to generate sequences for both display, analysis, and listening by simply constructing sequences of samples. For example, a first lenrth-8 cycle might be [1 p_1 p_1 p_1 -1 p_1 p_1 p_1] for a starting value of $p=p_1$. The next cycle could be [1 p_2 p_2 p_2 -1 p_2 p_2 p_2] for a second value $p=p_2$. Note that here we are simplifying the evolution of the parameter p by assuming it does not change during any one cycle. This should suffice. Further, in our examples, the parameter p will not change greatly on a cycle-by-cycle basis. It is clear that more complicated contours for the parameter p are fairly east to achieve.

As a first example suppose we have the parameter p changing very slowly from cycle to cycle and that it only varies once from p=+1 to p=-1 during what we view as a modulation process. Thus, for a length 8 generating sequence [1 p p p -1 p p p] we start with [1 1 1 1 -1 1 1 1] and this is followed by [1 0.999 0.999 0.999 -1 0.999 0.999 0.999] and then with [1 0.998 0.998 0.998 -1 0.998 0.998 0.998] and then after 2000 concatenated sequences (16,000 samples) ends with [1 -0.999 -0.999 -0.999 -1 -0.999 -0.999 -0.999]. Here is the related Matlab code:

```
dpsig=[]  % double pulse signal
p=1
for n=1:2000                % 2000 cycles of length 8, dynamically changing p
   dpsig=([dpsig [1 p p p -1 p p p]]);
      p=p-.001;
   end
   dpsig(1:50)              % first 50
   dpsig(16000-50:16000)    %  last 50
   sound(dpsig,2000)        % sound of full sweep - approx 8 seconds
   pause
```

Clearly this is far too much data to display, but we can listen to it. In fact, this is about 8 seconds of sound and resembles what we are familiar with by setting up a waveform and turning a panel control knob slowly. We hear a demo of a dynamically varying spectrum. Above we suggested that a “spectral dynamic” be measured according to that from a flat VCF being about a 1. In this same scaling, a sweep with a high-Q filter (non-flat) might be a 2 or 3, and a FM generated tone might be a 4 or a 5.

VERY subjectively, PWM might be about a 1.5, and spectral dynamics of the double-pulse shaper is about the same. The similarity of the double-pulse shaper to PWM is not really a surprise. There is a difference. The double-pulse shaper has less of what we might call an “edge” than PWM. It sounds a lot more like filtering than PWM does. This is not to suggest that one is better than the other in any sense.

Of particular interest is the case where p goes through zero – the immediate region about p=0. Nothing remarkable happens although we do hear the change of timbre become hollow, as we suggested, and then having passed through, emerges again fuller.

The experiment just above of a single sweep is of the nature of an envelope (once per tone) as described in Fig. 3. While we speak of “timber modulators” we first think of them as producers of variable waveshapes (but essentially static once a knob is set). There is little in the sense of a modulator in this case of a single, one-directional sweep (p=+1 to p=-1). This is something we can easily address by continuing to adjust the parameter p. We can turn it around and go back up, then down again, and so on. We will likely also need to increase the step size in p to get a high enough frequency for the modulation and enough modulation cycles to get something to listen too.

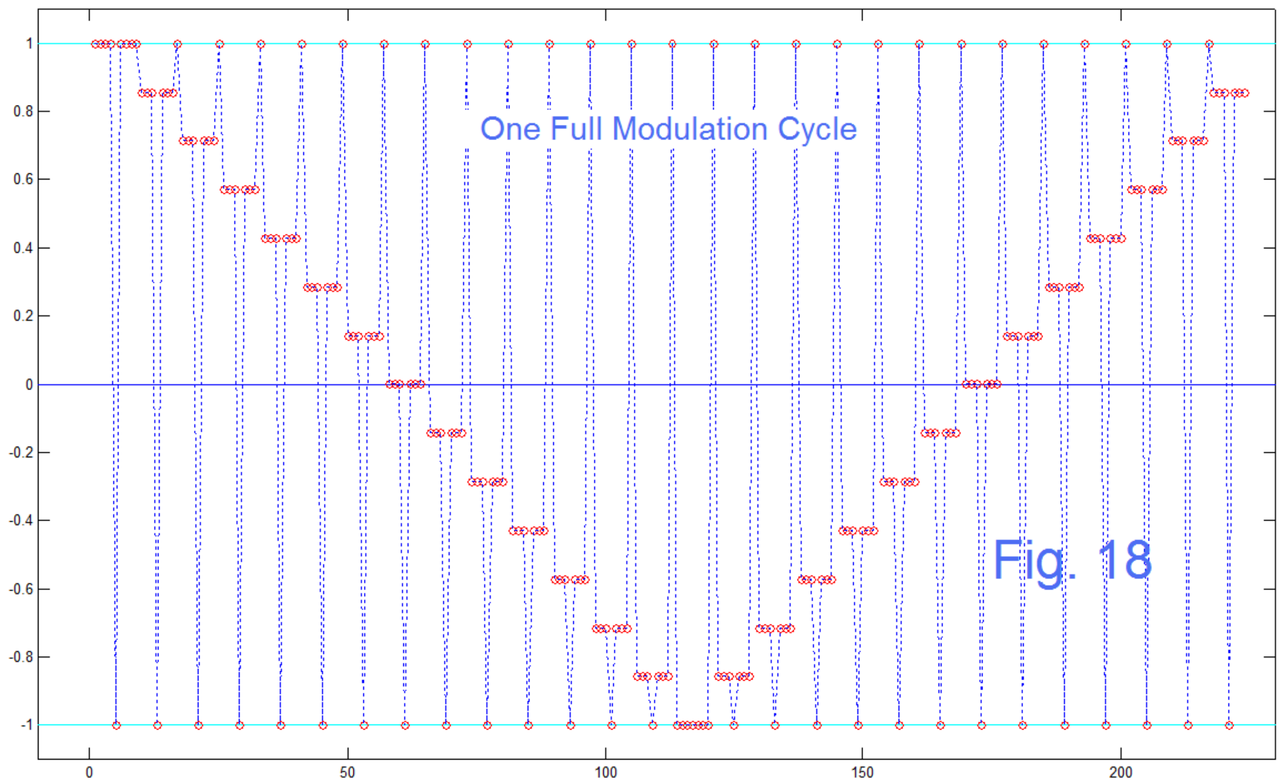


Fig. 18 shows one full modulation cycle of a total of 15 used. Here p is modified each full length-8 sequence by 1/7 instead of by 1/1000. This gives 14 steps down and 14 back up for each modulation cycle, or 8 x 28 = 224 samples per cycle. Because it is a bit difficult to see the details, Fig. 19 shows some details of the start of a modulation cycle (top) and the turn-around at the bottom (middle of cycle).

So Fig. 18 is a full cycle of our test signal. The full signal is 15 such cycles long, or 15 x 224 = 3360 samples, and was played back at a 5000 Hz sample rate. The sound was thus just over half a second and was a typical complex modulated sound. This was just one test.

More interesting than the sound at this point is the spectrum as obtained using the FFT. The FFT of a single cycle as in Fig. 18 tells the full story, approximating the Fourier Series as well. As always, it is a good idea to take a look at the time waveform to guess what the FFT should look like. WELL, from Fig. 18 it certainly looks as though we are dealing with a triangle-like waveform – at least in steps and we could reduce these by calculating p for each sample, not just for eight samples in a group. Indeed, the triangle is present. Less striking, we should not forget the values at +1 and at -1! One of every four samples is at the +1 or -1 limits. What do we see, what don't we see, and why?

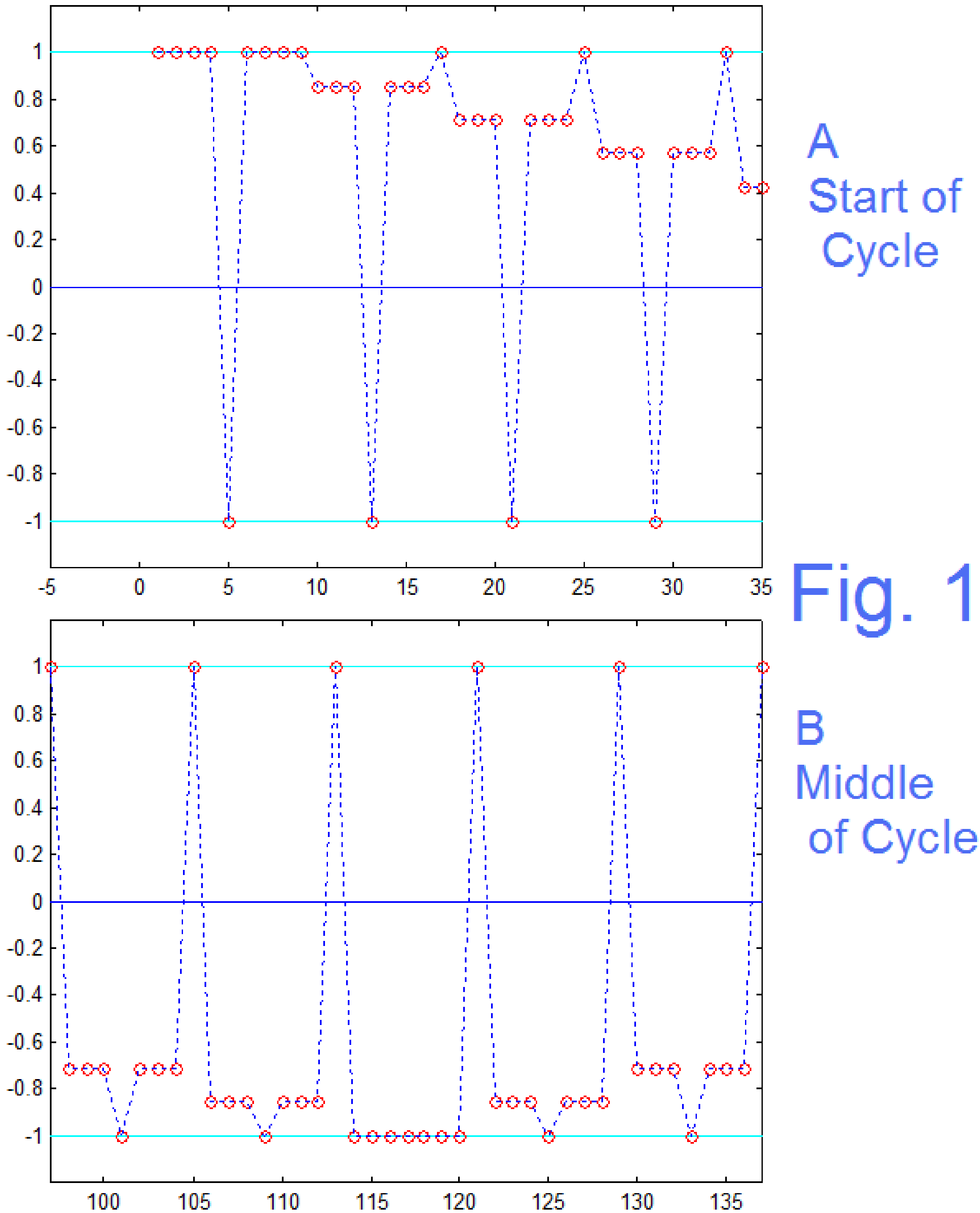
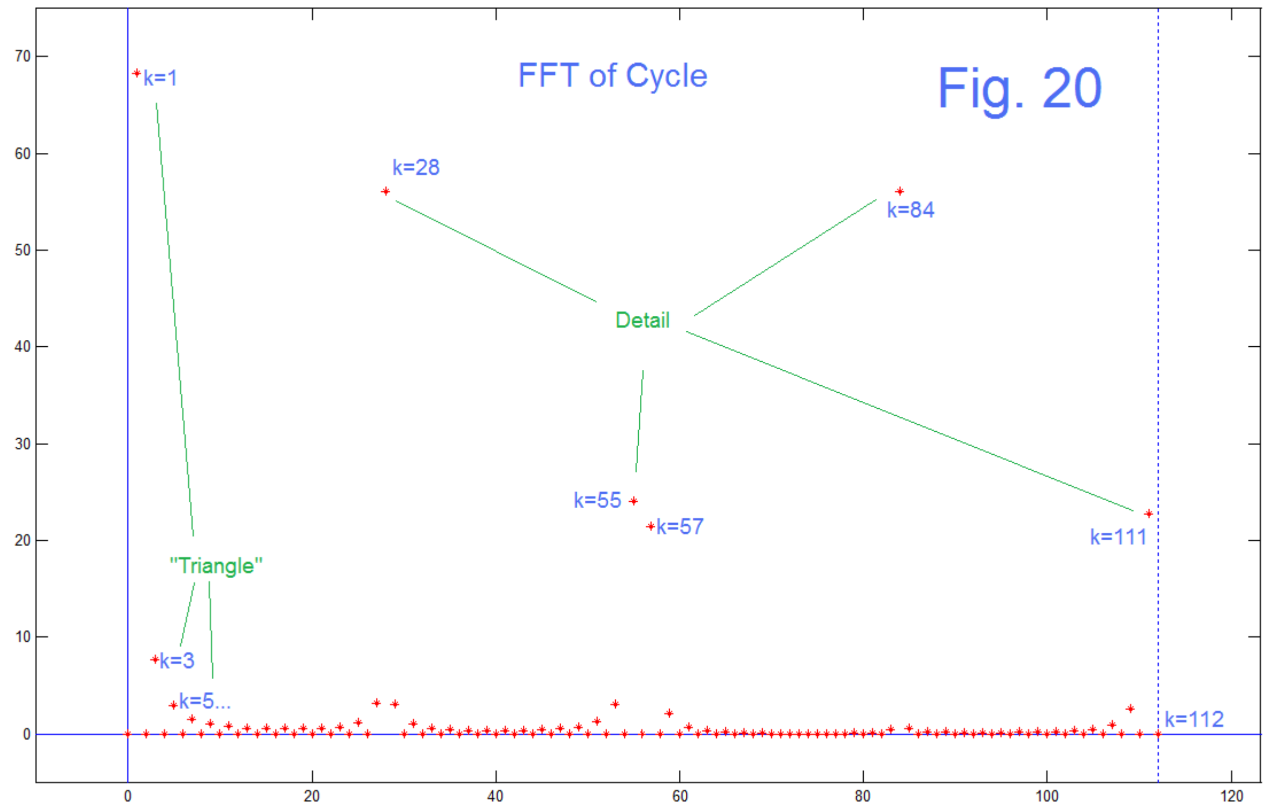


Fig. 19

We see in the FFT, Fig. 20, first of all, NO dc component. Given that the value of p is a substantial local bias, indeed the real local basis for seeing a triangle, note well that it is missing simply because it averages to zero over a full cycle. This makes sense. Less obviously, perhaps, the even harmonics are <u>mostly</u> (but not all) missing, again for reasons of averaging.



At this point, we can identify two things about the FFT that are interesting. Of course the FFT is linear, so we look at the plot as possibly explainable using superposition of things we do understand. Fig. 20 shows the overall "triangle" as a period 224 sequence which would have energy at k=1, and all odd harmonics. So we see what may be the triangle for small values of k. In fact, the k=3 and k=5 components are roughly the anticipated $1/k^2$ amplitudes. So perhaps the period 224 material is understood.

The second part of the superposition would be the length 8 structure due to the [1 p p p -1 p p p] sequences (with p changing). This structure repeats 224/8 = 28 times in the modulation cycle (Fig. 18). Thus we anticipate energy at k values that are multiples of 28. These we see at k=28 and k=84, but not at k=56 and k=112. But we have energy smeared about all four of these values, and indeed about k=0 combining with the triangle. The exact cancellation at k=56 and k=112 is due to cycle averaging – note that the energy in the values surrounding these values is substantial (k=55, 57, 111, and indeed at k=113, etc.). Thus Fig. 20 shows the "triangle" and the length-8 "detail".

It may be possible to understand the spectrum in terms of modulation sidebands. The "carrier" components are at k=0, 28, 56, 84, and 112, due to the sequence details. These are "modulated" by the changing p, which cycles once per modulation cycle, hence the components spaced at ±1, ±2, etc. as "sidebands". This seems like a correct mode of analysis, but one which we have not attempted here. Likely the understanding based on the numerical generated result here will have to do. Anyone ambitious?

TIMBER MODULATOR EXAMPLE 2
– Bill Hartmann's Symmetrized Ramp (SR)

Above we discussed the various conversion schemes for forming waveshapes from one another. The conversion scheme from triangle to sawtooth was necessary when one starts with a "triangle-core" VCO (the integrator/Schmitt-trigger type) which is easy to achieve if we have the corresponding square wave in the proper phase. That is, the square was the sign of the derivative of the triangle, (but which was inherent in the VCO). None the less this gave a sawtooth of twice the frequency until such time as we added in the square in the right amount (Fig. 6 and Fig. 7). In our myopia, we tended to look on this as a matter of doing things exactly right, or else we fail. Bill Hartmann on the other hand had the vision to recognize the conversion as one particular case of a more general process [7b] to form a "symmetrized ramp" (SR) of which the sawtooth was just one special case. This SR was an idea we then used in our triangle-core VCOs [7c] which was a very economical addition, and probably would have been more widely used were it the case that the "triangle-core" VCO approaches were as popular as the "sawtooth-core" counterparts.

Here in using Bill's SR as a second example to Ian's double-pulser, we note several differences in coverage here. First, there is the extreme circuit economy of Bill's addition. Secondly, while we apparently neglected the full analysis of Ian's device, when we looked we found most of the analysis of Bill's idea in our pages [7b, 7c, and 7d].

As proof that the SR is a continuation of the triangle-to-saw process we show Fig. 21, which is an exact continuation of Fig. 7 [12]. Here the top four waveforms are the conventional shaper. The waveforms below show Bill's more direct approach. That is, the waveform of (21e) is obtained by offsetting the triangle so that it just touches zero, but it otherwise always positive. This means that it has twice the amplitude, but this is no problem (just a scaling) and the square wave can multiply this to give (21f), which is identical to (21d). Indeed we can write:

(21d) = ½×(21a)×(21b) + ½×(21b) = ½×(21b)×[1 + (21a)]

and also:

(21e) = (21a) + 1

so that:

(21f) = ½×(21e)×(21b) = ½×(21a)×(21b) + ½×(21b) = ½×(21b)×[1 + (21a)]

so the result is the same sawtooth. The difference is in the circuit implementation. Bill's method borrows the offset circuitry normally associated with the PWM, so that the sawtooth

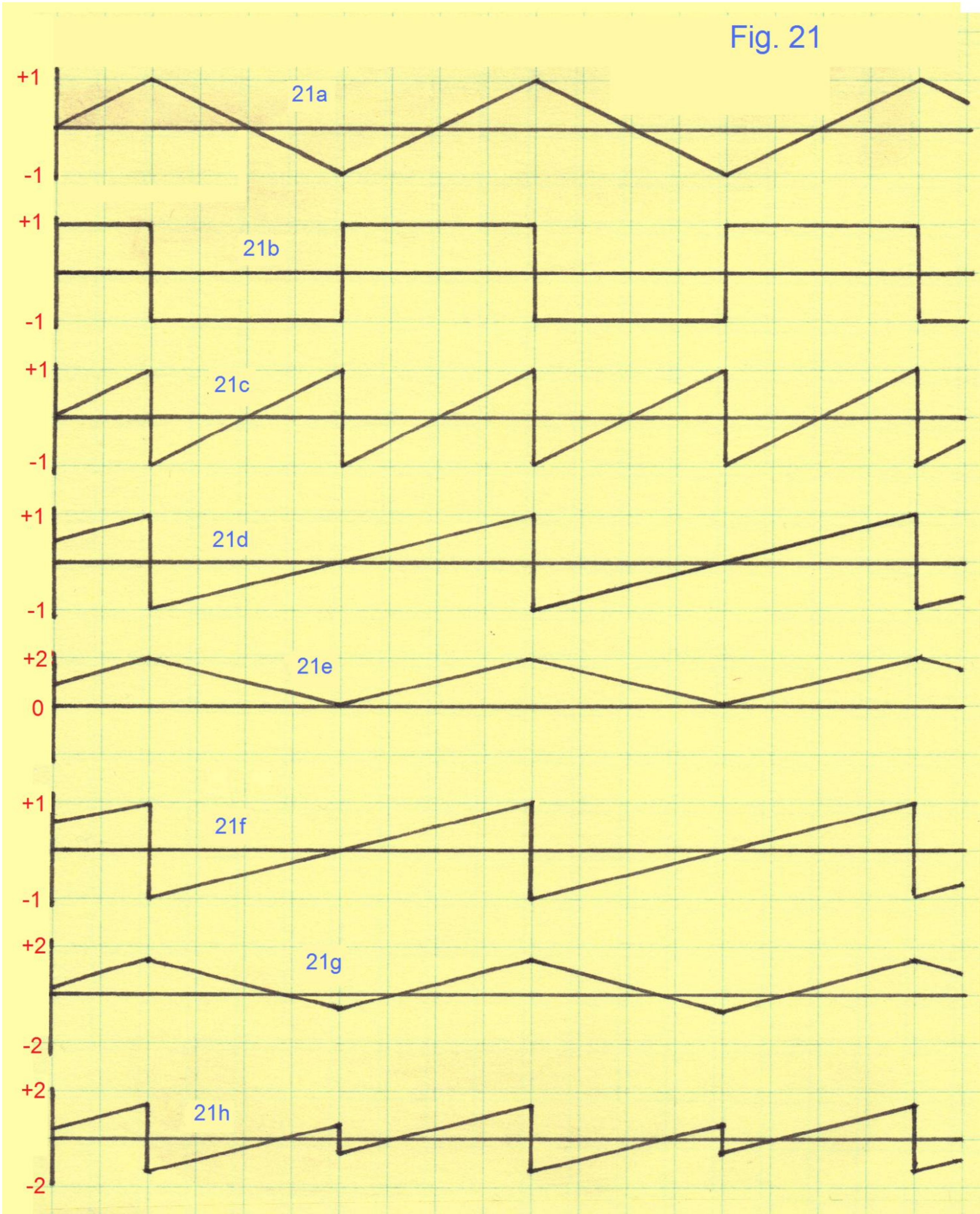


Fig. 21
+1
21a
-1
+1
21b
-1
+1
21c
-1
+1
21d
-1
+2
21e
0
+1
21f
-1
+2
21g
-2
+2
21h
-2

output is a SR (21h) with the “normal” sawtooth a special case (21f). Fig. 22 shows the circuitry that supports both PWM and SRM (Symmetrized Ramp Modulation).

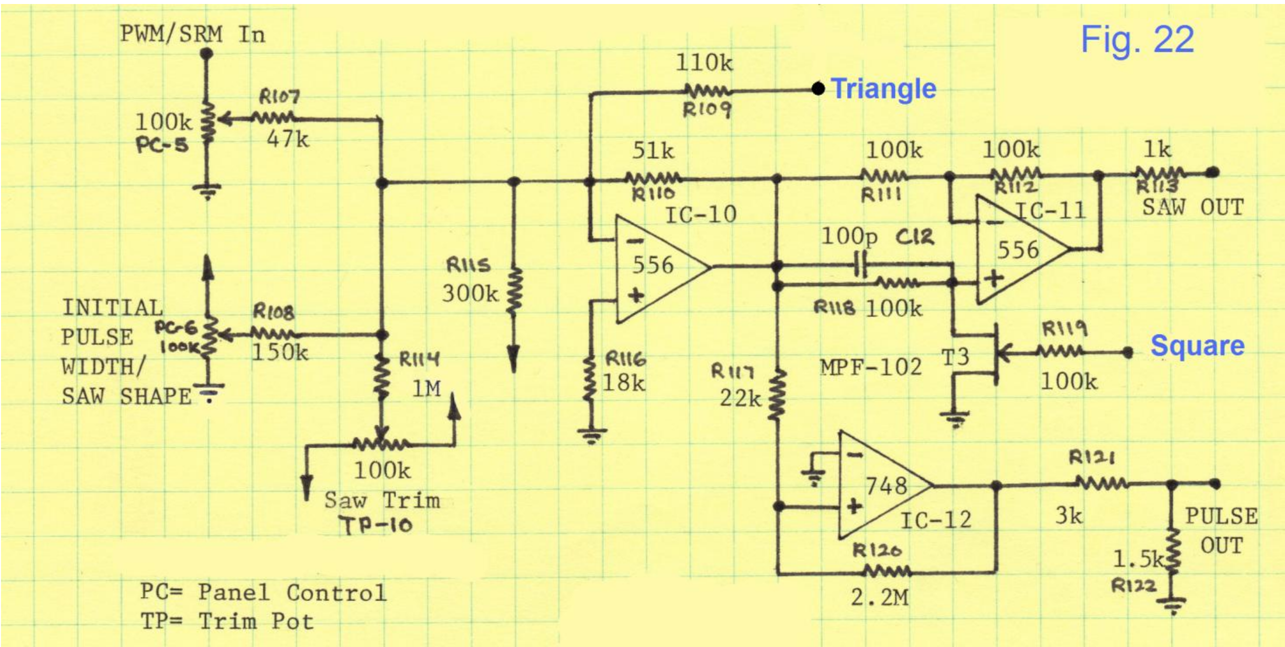
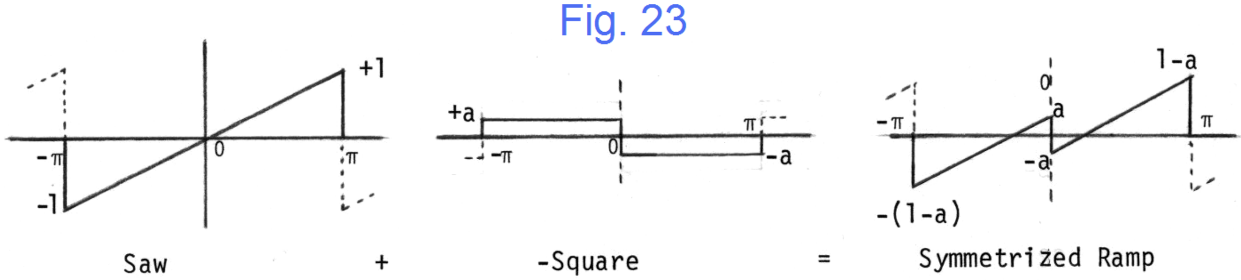


Fig. 22

Faced with the need to obtain a frequency description of SRM we can consider doing the Fourier Series, approximating with a FFT, or hope that someone else did it already. Here it came as a pleasant surprise that (1) we already did it, and (2) it is very simple in this case anyway, due to superposition. In particular, from what we have discussed above, we know that the SR is the sum of a sawtooth and a square. Now, a square has only odd harmonics. A saw has all harmonics. As we change the amount of the square wave (see Fig. 23) we change the odd harmonics but the even harmonics do not change. This was discussed in EN#121 back in 1981 [7d]. What a happy situation overall. Further, most of us have doubts (at least worries) about our “old work”. Here with the new tool of the FFT we can verify the result.

Fig. 23



Here the symmetry about zero for the saw and the square (and consequently, of the SR) means that all Fourier Series components can be taken as sines and we only need sum the coefficients. The even frequency coefficients will be constants in the summation and we will look for a case where the odd frequencies go to zero. This will be when the square is of amplitude a=0.5, and the result is a double frequency sawtooth, which has (relative to the original frequency) only even harmonics.

The saw of Fig. 23 is entirely "textbook" and its Fourier Series is:

$$f_{SAW}(t) = (2/\pi)\,[sin(t) - (1/2)sin(2t) + (1/3)sin(3t) - (1/4)sin(4t) + \ldots..]$$

while the square is textbook except for an inversion and scaling by a factor a:

$$f_{SQUARE}(t) = (-4a/\pi)\,[sin(t) + (1/3)sin(3t) + (1/5)sin(5t) + (1/7)sin(7t) + \ldots..]$$

and these two can be added to give the SR:

$$\begin{aligned} f_{SR}(t) = \; & (2/\pi)(1 - 2a)\,sin(t) - (1/\pi)\,Sin(2t) \\ & + (2/3\pi)(1 - 2a)\,sin(3t) - (1/2\pi)\,Sin(4t) \\ & + (2/5\pi)(1 - 2a)\,sin(5t) - (1/3\pi)\,Sin(6t) \\ & + (2/7\pi)(1 - 2a)\,sin(7t) - (1/4\pi)\,Sin(8t) + \ldots\ldots \end{aligned}$$

Here is the Matlab code that gives an excellent verification by FFT (the program also illustrates the use of the FFT for FS estimation [14]:

```
%  EN228c      Test SRM
a=0.5    % choose parameter a
%  Compute Fourier Series Coefficients (s) from Closed-Form Formulas
for k=1:2:20
   s(k)=(2/(k*pi))*(1-2*a);
   s(k+1)=-(2/( ( (k+1)*pi)) );
end
s
% one cycle for FFT
x=-a:1/1000:1-a-0.001;
x=[x -(1-a):1/1000:a-0.001];
plot(x)
X=fft(x);
FFTX=-imag(X(2:20))/1000      % look at first F.S. by FFT approximations
```

This result for $f_{SR}(t)$ shows that all odd harmonics are multiplied by (1-2a) while the even harmonic amplitudes are constants independent of the parameter a. This we illustrate in Fig. 24a, from [7d], and contrast with the much more "active" case (note notches) of PWM in Fig. 24b. We note that both SRM and PWM in the circuit of Fig. 22 both are parameterized by the same panel controls.

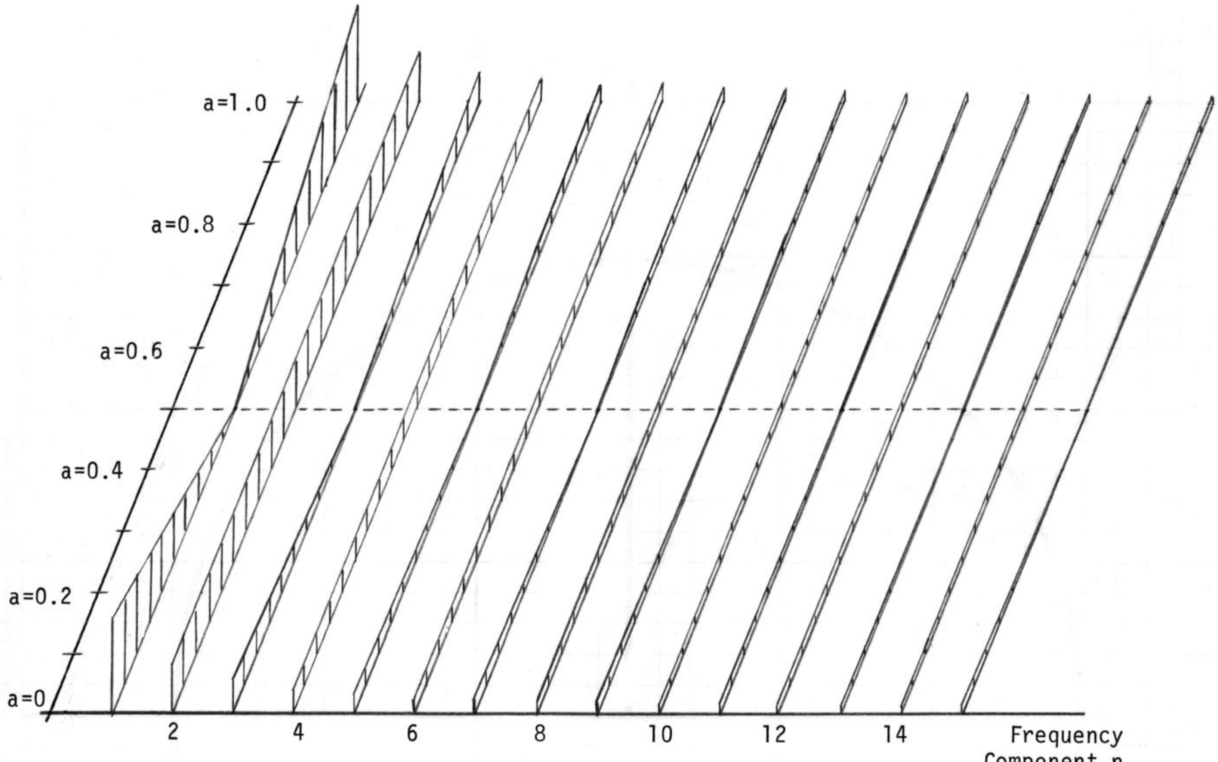


Fig. 24a Spectrum Map of Symmetrized Ramp Modulation

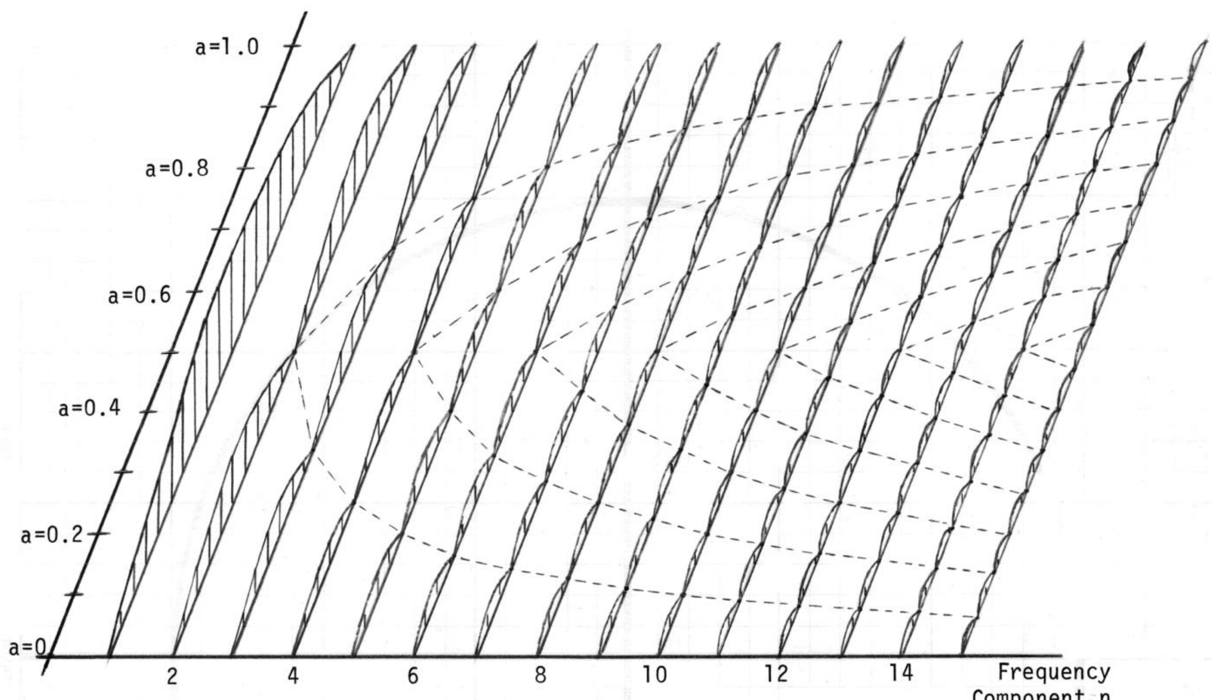


Fig. 24b Comparison Spectrum Map of Pulse-Width Modulation

Fig. 25 is in preparation for the use of the FFT to compute the spectrum of SRM. We will be computing the F.S. of a SR as the sum of a saw and a square. So here we approximate the saw and square by 100 samples, and will allow that the F.S. can be obtained from the FFT of these samples, here for the first 20 coefficients. Thus Fig. 25

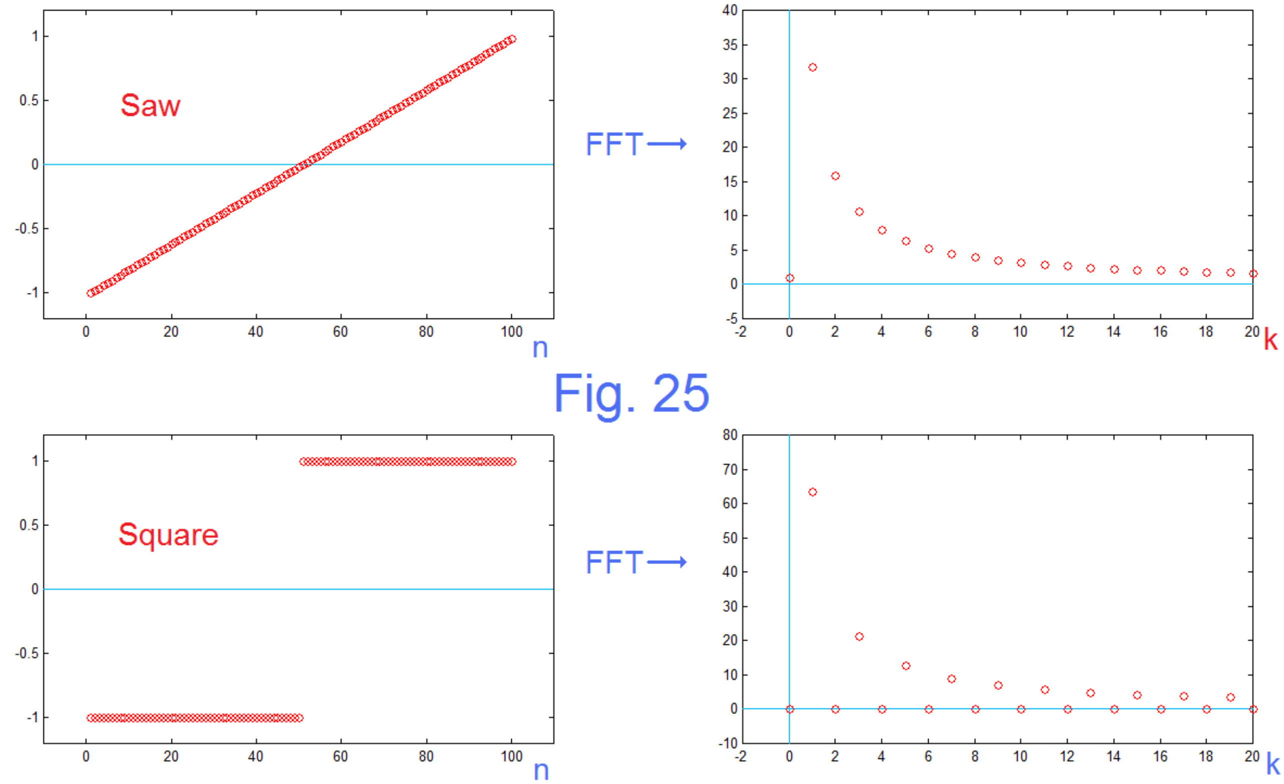

Saw
FFT ⟶
Square
FFT ⟶
n
k

Fig. 25

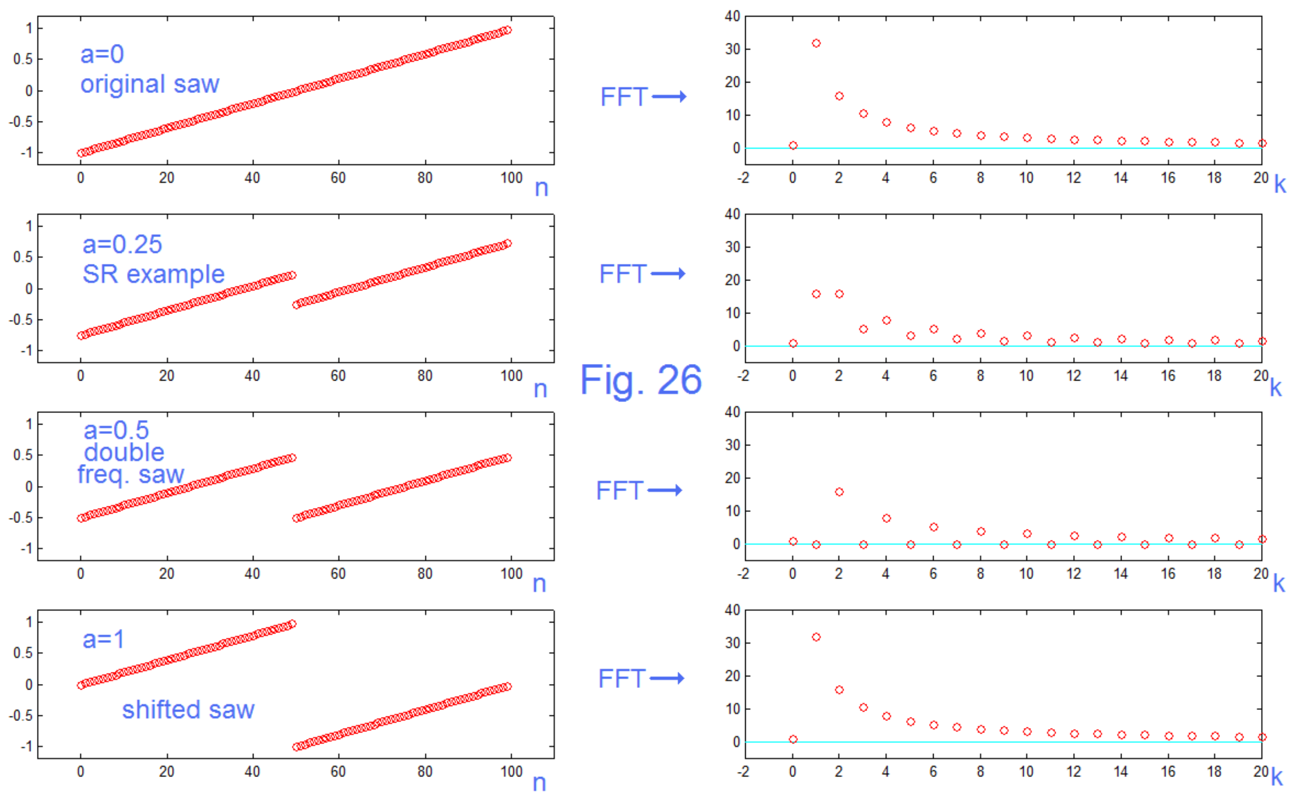
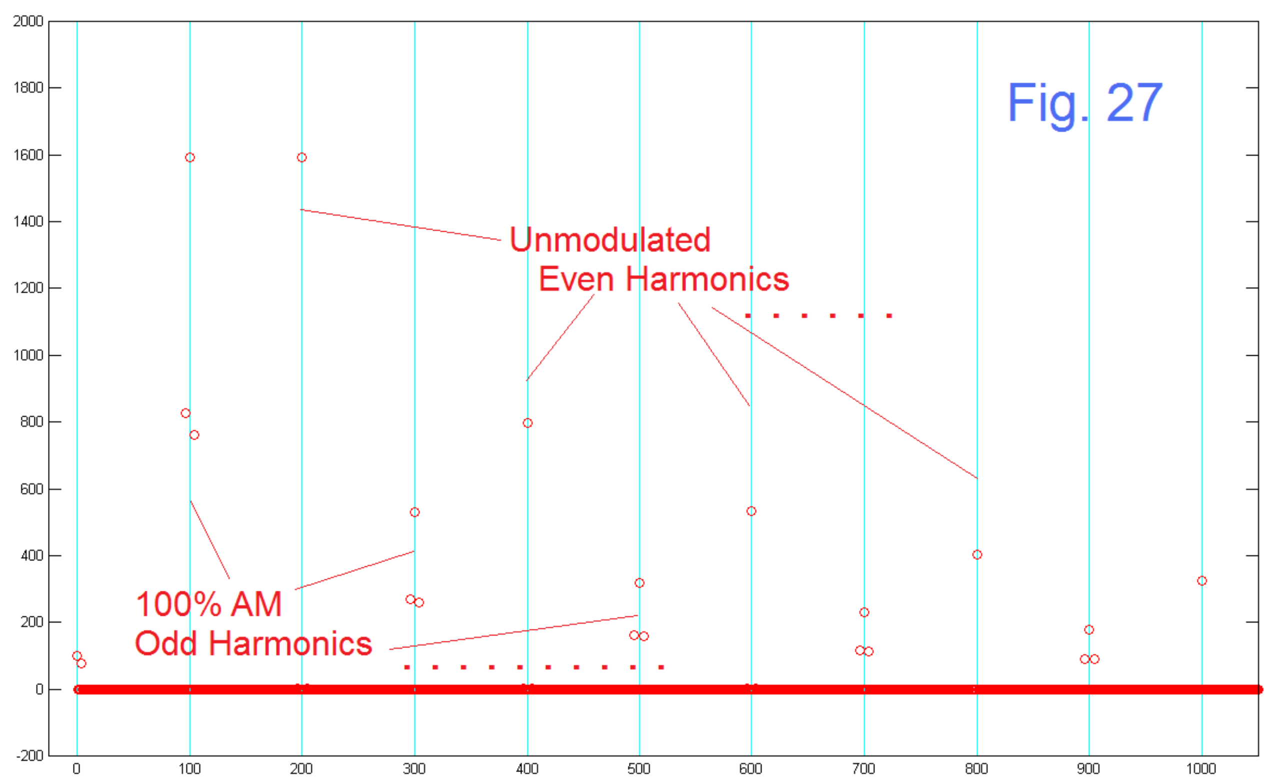

a=0
original saw
a=0.25
SR example
a=0.5
double
freq. saw
a=1
shifted saw
FFT ⟶
n
k

Fig. 26

corresponds to the two signals that make up the SR. Fig. 26 shows the coefficients of the SR for values of a=0, a=0.25, a=0.5, and a=1. Here we show on the left a 100 sample version of the four example SR waveforms, and on the right, the FFT magnitudes. Thus the a=0 case of Fig. 26 is exactly the original sawtooth. The a=0.25 case starts to be a SR. We see the splitting into two segments, and the odd harmonics are starting to fall off. The even harmonics remain constants. The case of a = 0.5 is the special case where the frequency of the sawtooth doubles. Note that the odd harmonics have become zero. The even harmonics are now in the ratio of the sawtooth, with half the amplitude. The final example is a=1 (having here limited a to the range of 0 to +1, which is not a rule which can't be broken!), and goes be to a sawtooth that is now shifted by half a cycle. (This is a neat F.S. example for teaching.) (This shifted saw we will compare to the continuous shifter in our third example to come.) The general result is Fig. 24a as noted.

Here we come to a very interesting example. It has been a task here to show the spectra as parameter driven modulations. As such, we represent spectral evolutions as successive "snapshots" in sequence. Here we have an opportunity to find a result that is familiar from ordinary Amplitude Modulation (AM). When we look at the F.S. coefficients for the odd harmonics of the SR we note that they are all driven by a factor (1-2×a) and if we make the parameter a a function of time, a = a(t), we can achieve the case of (for example) 100% AM by choosing a(t) = (1/2) + (1/2)×Sine where Sine is taken to be a modulating sinewave between +1 and -1 [18]. That is, we set (1 - 2×a(t)) = 1/2 +(1/2)×Sine, or a(t) = 1/4 – (1/4)×Sine. This should result in a "carrier" with two adjacent sidebands of half the carrier amplitude [18]. Plugging this in for a sequence of cycles which are sequenced for a series of values of a(t), we compute a full modulation cycle and take the FFT of the whole (see code in Appendix).



This done, we obtain the result of Fig. 27. We have exactly 100 sawtooth cycles for a full modulation cycle, so the values of k in Fig. 27 are 100 times larger than the actual harmonics. We see, remarkably (!), pretty much what we set out expecting to find – the odd harmonics are 100% AM (central carriers with two sidebands at levels approximately 1/2 for each) and the even harmonics are unmodulated. (As previously, we see simplifications due to averages over a full modulation cycle.) All and all, a lovely result.

TIMBER MODULATOR EXAMPLE 3 – Ludwig/Hutchins Sawtooth Shifter

For our third (and final – for now) example we turn to the simple sawtooth phase shifter that is the basis for the popular Multi-Phase Waveform Animator (MPWA) [8, 9] which is a bit different from the first two examples. First, it does not change the spectrum in a significant manner when changed at a slow rate (it is not a static waveshaper – always looking like and sounding like a sawtooth unless modulated at an audio rate). Secondly, it was employed in parallel low-frequency-oscillator mode as an animator. We note here that we include it here as Ian was inspired to pursue it and to suggest the further study here.

Fig. 28 reminds us of the circuit structure and Fig 29 shows the various waveshapes. The input parameter is the control voltage V_c, which modulates the phase. In a manner quite similar to ordinary FM, the spectrum is not significantly modified by V_c until such time as the depth and frequency result in significant (generally not even harmonically placed) "sidebands". Or, as with the MPWA, when recombined in parallel components.

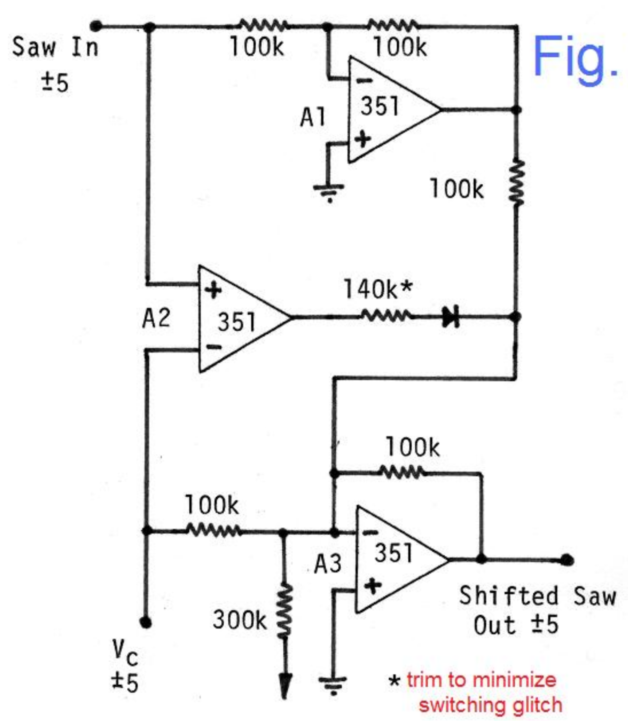
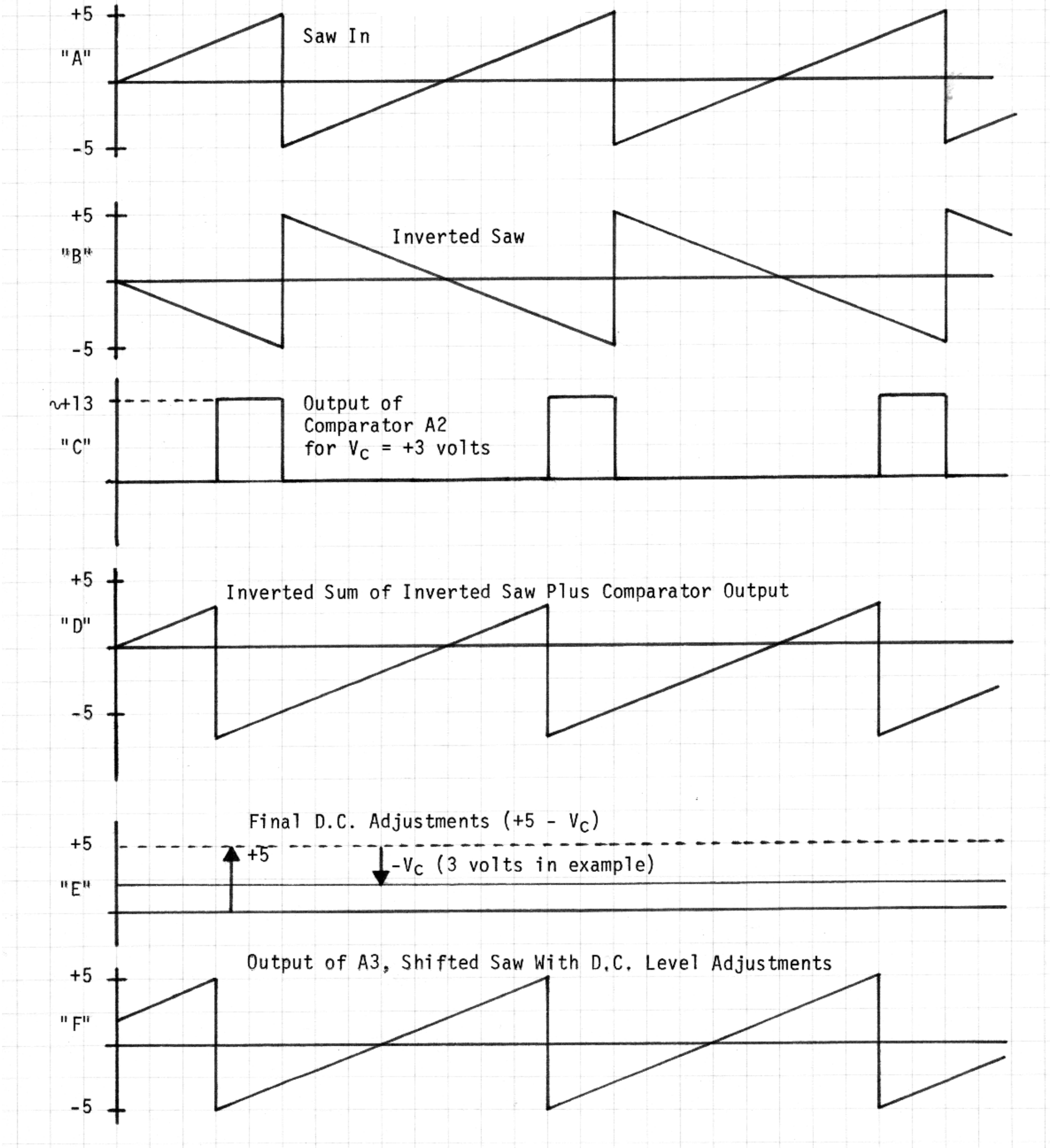


Fig. 28

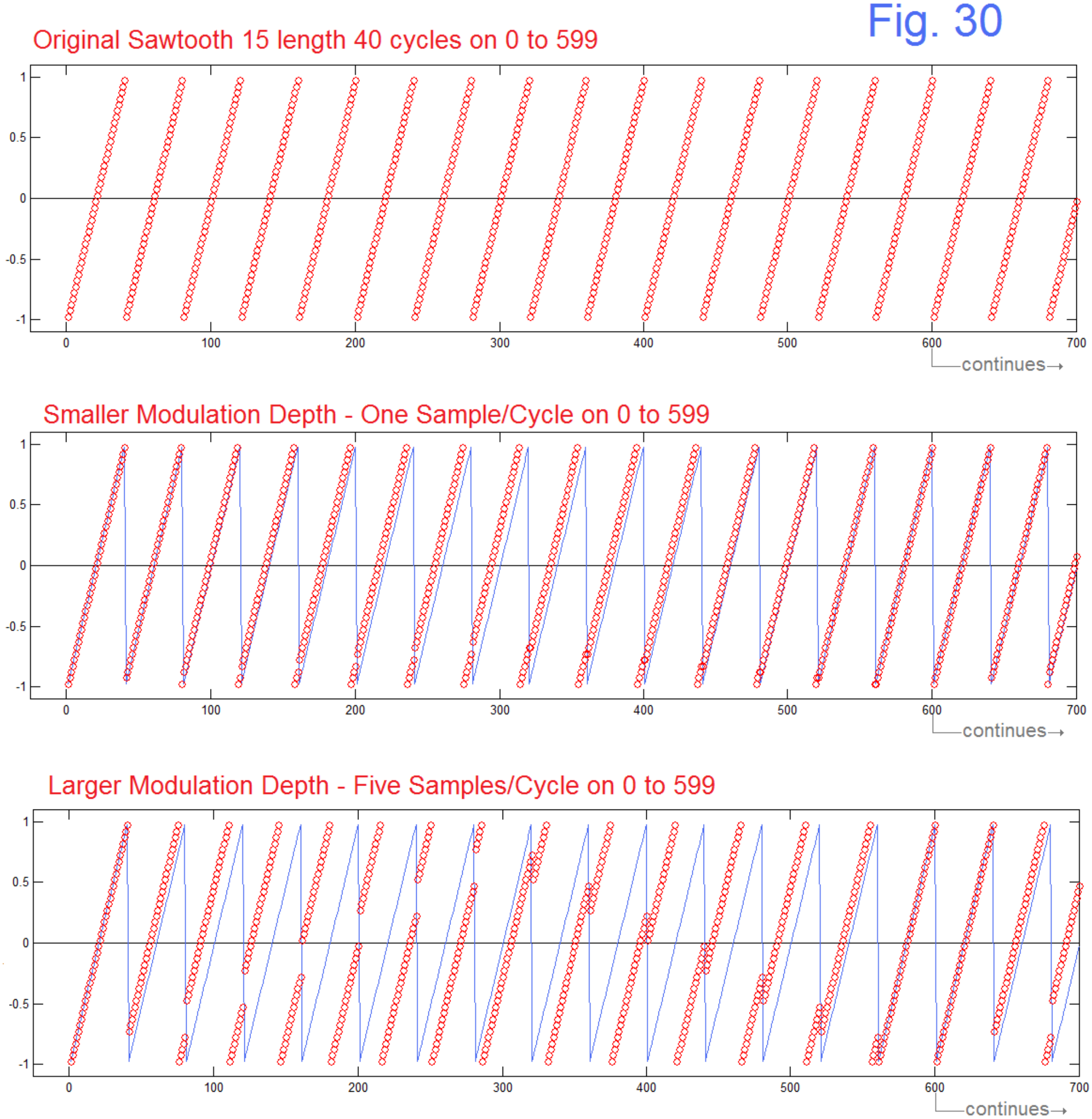
Fig. 29

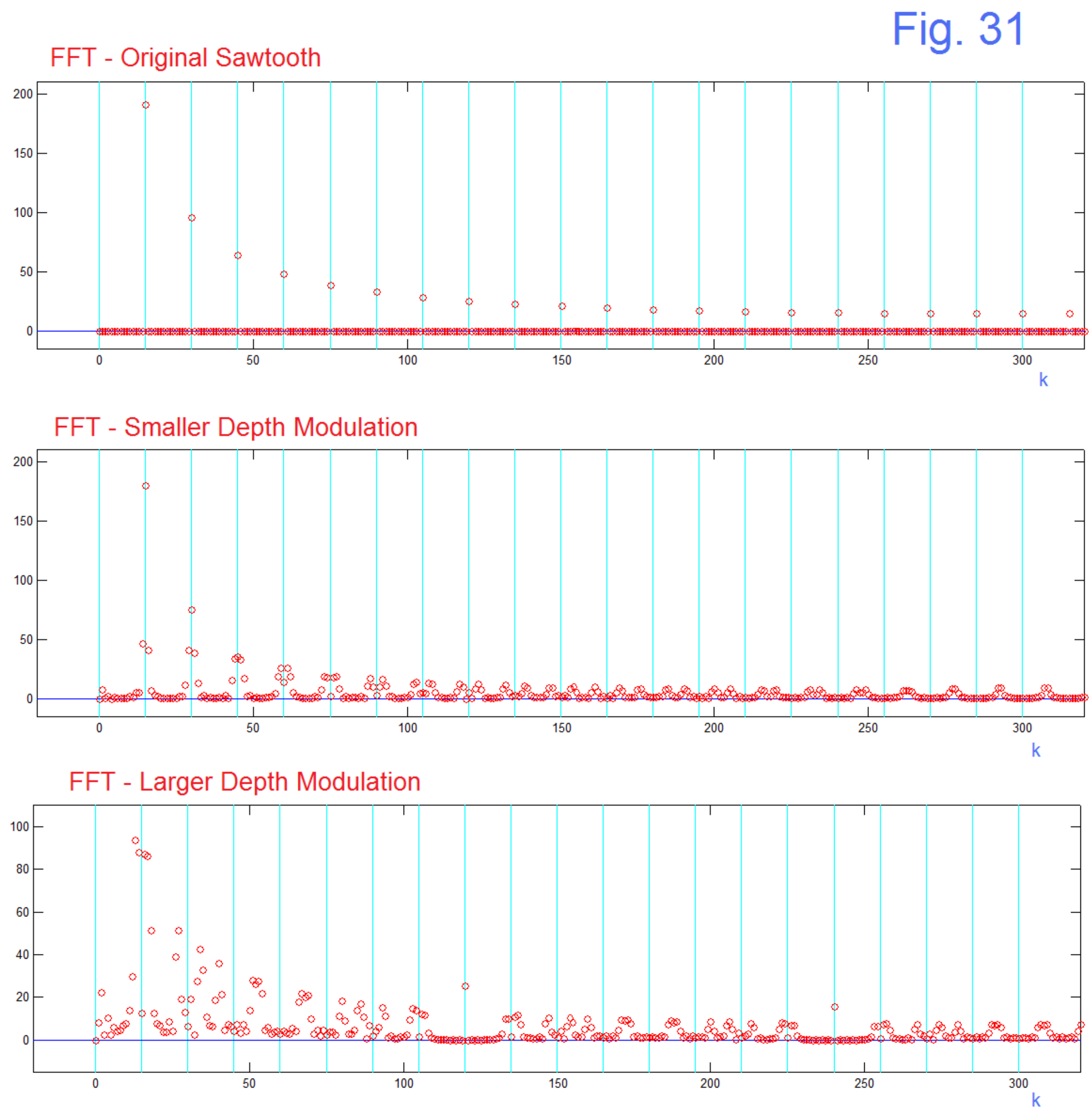


The phase modulation (PM) problem is less familiar than the FM problem, but quite similar [19], sometimes being treated as a combined subject : "angle modulation". Most simply, if a sinusoidal waveform is progressing along and we expect at a time $t+\tau$ the value to changes from $x(t)$ to $x(t+\tau)$, but instead find a value $x'(t+\tau)$, we might well attribute the unexpected change to a shift from the expected frequency or to a different phase (or a different amplitude for that matter) . In consequence, we might anticipate here something that looks most like FM – i.e., sidebands.

In the manner of our lazy first look at issues here, we can consider a phase shift as a cycle-by-cycle adjustment of starting point of a sawtooth. Here (Fig. 30, top panel) we first construct a sawtooth of 40 samples from -0.975 to +0.975 incremented at intervals of 0.05. We can repeat this exactly for a standard sawtooth waveform. In the phase shifted cases, we do not start at -0.975, but at some later value. When we reach +0.975, we wrap around to -0.975 and continue for a total of 40 cycles. In the modulation case, we have in mind a series or starting points (probably in sequence) with spacing one or more samples each cycle. Fig. 30 (middle panel) shows the case where new cycles start with a jump forward of one sample for the first half and then jumps backward by one sample for the second half. Note that this is seen in the bottom 20% of the middle panel. For easy reference, the original sawtooth from the top panel is under-plotted as a continuous waveform in blue. Note well that the modulation moves the samples “early” for the first half, and “late” for the second half, a phase modulation looking like a frequency modulation. The bottom panel of Fig. 30 is the corresponding case where the phase jumps are by 5 samples rather than just one. This greater modulation depth uses most of the available range.

Fig. 30





Here we turn attention to the frequency domain in search of a complex spectrum and likely a sideband structure. As in the examples above, we employ the FFT so will look at samples over a full modulating cycle (600 samples here) and relay on averaging. In the 600 samples, we have 15 cycles of the sawtooth "carrier", so we look for FFT energy at values of k that are multiples of 15. In the top panel of Fig. 31, we have exactly energy at k=15, 30, 45…. up to 300, at which point the FFT magnitude reflects, and is not plotted. This is the familiar unmodulated sawtooth as seen several times above.

The modulated cases, the middle and lower panels of Fig. 31, do in fact show sideband-like splitting of the sawtooth components. This looks right (and is right if we rely on the

FFT), but we lack closed-form equations. At the same time, the waveforms as in Fig. 30 are far too short for a good listening test. However the modulation cycles are periodic with period 600 and we can splice enough of these together for thousands of samples and seconds of listening. Aurally, ibt is consistent with expectations that the middle case (smaller depth modulation) is a typically modulated sound, while the lower panel (larger depth modulation) is a rather aggressive spreading of the spectrum similar to the general cases of FM. It works!

The FFT's are interesting. For the middle panels, left side (k=15, 30, 45, 60 or so) we see multiple sideband pairs, carrier centered. On the right side note that the sideband spreading seen starting on the left side has progressed to the point where the carrier is apparently gone, with sidebands from adjacent carrier positions bumping each other. Here no good explanation is at hand. (Perhaps if one recalls how the higher-order Bessel functions start up slowly, this may suggest something.) The bottom panel of Fig. 31 (larger depth) is quite a puzzle and rather hopeless except for guidance of the blue lines we have added at multiples of 15. For example, why strong components at k=120 and k=240 with so much empty space in the adjacent regions? Lots of regions to explore and explanations to seek. Note that digitally, we can continue into regions beyond the analog limits. It is clear that this phase shifter has offered a very strong modulator at very little hardware cost.

At this point I go back to Ian's email:

2/24/2016

"One we missed?

Well at least I did.

....

It just dawned on me that your MPWA uses the simplest phase shifter in the universe, so what happens if I modulate it at audio rates? My unit was build with non-dedicated PM inputs, so I just had to plug in.

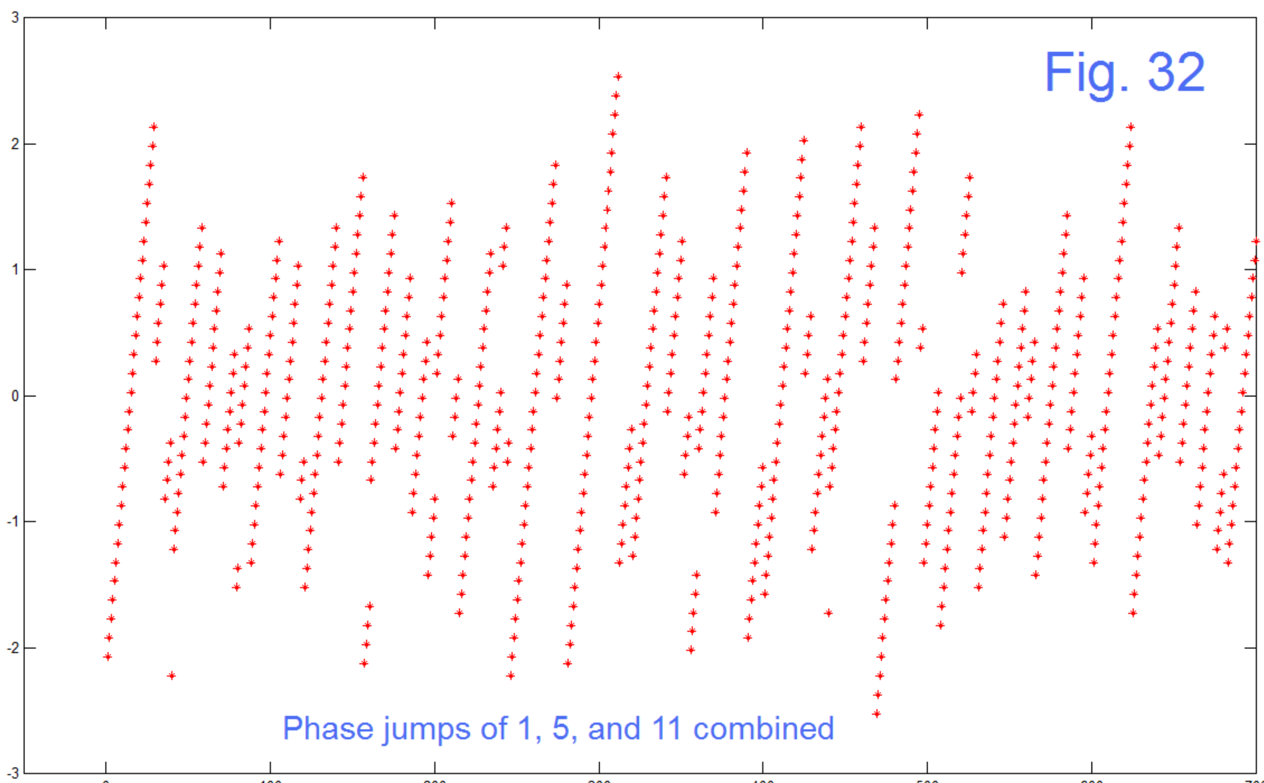
Modulating just one of the shifters works very well, with typical bright "FM"-like sounds. So that's an easy thing to tack onto a VCO to get linear PM. But then I tried using two of the shifters with separate audio inputs. I was expecting a lot of unpleasant noise, but instead I found a set of rich complex tones which were different from anything I have ever heard! A pleasant surprise. . . . . ."

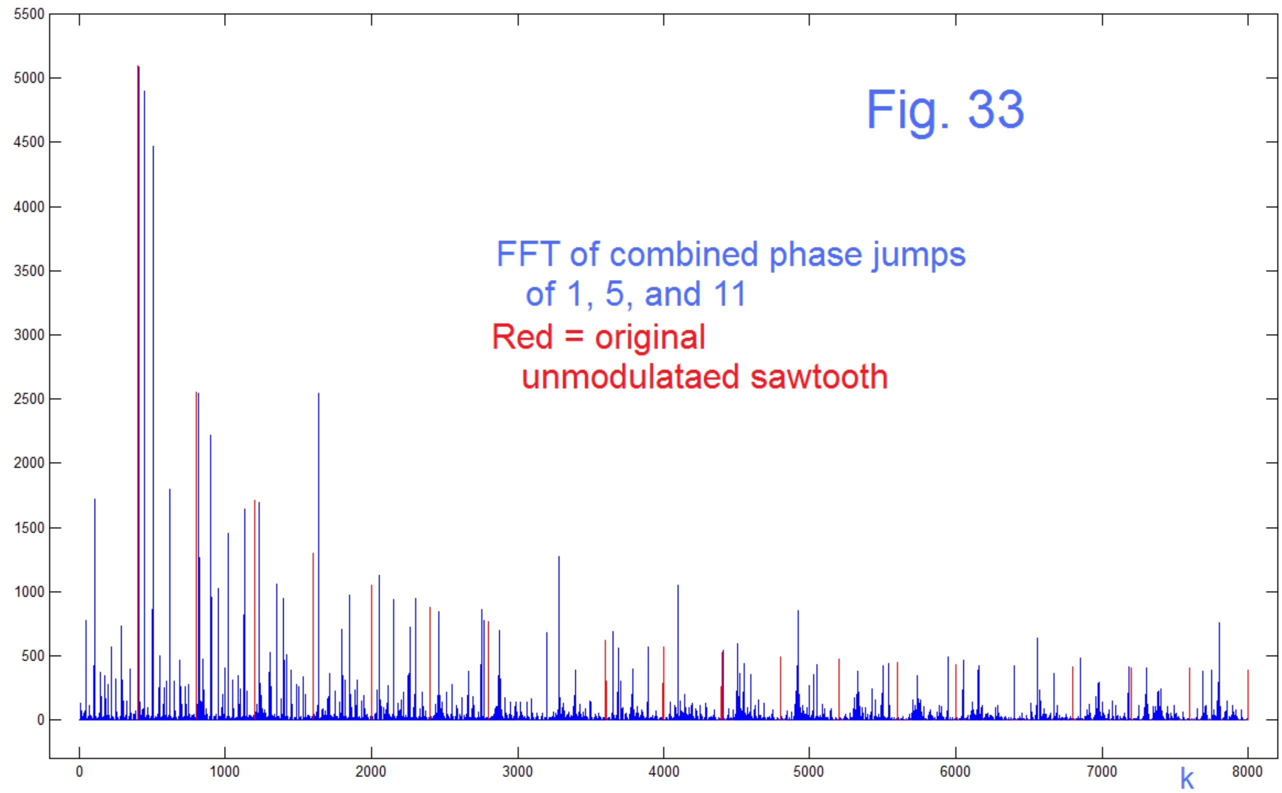
So at this point we are caught up with Ian – more or less. Ian found the 'typically bright "FM" –like sounds' (as we saw in Fig. 31). Going beyond that, we have always suggested the value of digitally investigating the dimensions and the parameter ranges of synthesis modules by "cut-and-paste" rather than by intensive construction efforts. (Not that we have followed our own suggestions of course!) Here we can write code that allows us to test waveforms generated in Matlab. The code below shows the potential of eight parallel channels (turned on/off with the amplitude vector ***as***. The phase jumps are contained in the parameter vector ***st***. The ***mod*** function is used to keep the segments in range (sawtooth rather than triangular reset). No comprehensive investigation was done here.

```
% en228e
clear
st=[ 1 5 11 34 15 16 27 28  ]
as=[ 1 1  1  0  0  0  0  0  ]

ss=-0.975:.05:0.975;
ss=[ss ss];
s=[];

for k=1:400
   snew=zeros(1,40);
   for cm = 1:8
     k1=mod(k*st(cm),40)+1;
     s1=ss(k1:1:k1+39);
     snew=snew+as(cm)*s1;
   end
   s=[s snew];
end
figure(1)
plot(s(1:700),'r*')
axis([-50 700 -3 3])
S=abs(fft(s));
figure(2)
plot([0:8000],S(1:8001))
axis([-200 8200 -300 5500])
SS=S;
sound(s,8000)
```





The example has three shifts: 1, 5, and 11 turned on (see code above). With three component shifts, we expect a result more complex than the shift of 1 or the shift of 5, and this we get, with a fairly metallic or machine-like sound (at least in the example). Fig. 32 shows the combined waveform (700 samples of it anyway) while Fig. 33 shows the FFT magnitude of all 16000 samples. Again – this is just an example run. Fig. 32 is a set of jagged segments all of the same slope, so something coherent is seen – and heard. But there are far too few results to comment much.

CONCLUDING COMMENTS

Here we have made the observation that by in large, we have been conservative in employing synthesis ideas. Not unexpectedly, we have tended to emphasize things that have been found to work, and thought too much that variations on the conventional wisdom must have been explored. A happy finding is that there is still a vast abstract "space" to explore. Also it was good to see that, at least in the three examples here, we were talking about simple circuitry (or program code) and not extensive and expensive hardware.

REFERENCES

[1] B. Hutchins, “Revisiting Pulse-Width Modulation”, ***Electronotes*** Volume 23, Number 216 July 2013 http://electronotes.netfirms.com/EN216.pdf

[2] B. Hutchins, “Reviewing the Current State Of Music Synthesis”, ***Electronotes*** Volume 23, Number 220 January 2014 http://electronotes.netfirms.com/EN220.pdf (Moog comment on page (2) or in Reference [3] below)

[3] “Moog Stuff Complied Here”, Electronotes Webnote ENWN-14 12/2/2013 http://electronotes.netfirms.com/ENWN14.pdf

[4] B. Hutchins, “Negative Frequencies – And Through-Zero Frequency Modulation,” ***Electronotes***, Volume 22, Number 206 December 2010 http://electronotes.netfirms.com/EN206.pdf

[5] B. Hutchins, “A Voltage-Controlled Oscillator with Through-Zero Frequency Modulation Capability, ***Electronotes***, Vol. 13, No. 129, Sept 1981 http://electronotes.netfirms.com/EN129.pdf

[6] http://electronotes.netfirms.com/TZFMNote.pdf

[7] Timbre Modulators

[7a] I. Fritz, “Reader’s Circuits” (Double-Pulse Waveform Shaper and two others), ***Electronotes***, Vol. 8, No. 72, Dec. 1976, pp 21-22

[7b] W. Hartmann, “Comments on the Triangle-to-Sawtooth Conversion Problem,” (“Symmetrized Ramp”), ***Electronotes***, Vol. 8, No. 67, July 1976

[7c] “ENS-76 VCO Option 2”, ***Electronotes*** Volume 9, Number 75 March 1977 pp10-14

[7d] B. Hutchins, “Fourier Series Results for Some Simple New Waveshapers”. Electronotes, Vol. 13, No. 121, January 1981, pp 5-13 http://electronotes.netfirms.com/EN121.pdf

[8] B. Hutchins, “Analog Circuits for Sound Animation,” ***J. Audio Eng. Soc***., Vol 29, No. 11, Nov. 1981, pp 814-820 http://electronotes.netfirms.com/AES4.PDF

[9] B. Hutchins, “A Sawtooth Driven Muli-Phase Waveform Animator,” ***Electronotes***, Vol. 10, No. 87, March 1978; Lester Ludwig & B. Hutchins, “Multi-Phase Scanning with Sawtooth Waveforms, Electronotes Application Note AN-73, Feb. 4, 1978.

[10] B. Hutchins, "Mathematical Analysis of Differential Amplifier Triangle-to-Sine Converter," ***Electronotes***, Vol. 9, No. 82, Oct. 77, pp 5-17 [**Reality Note 1:** Here we showed that the drive level for the lowest THD was 79 mV. Shortly after writing this I was visiting Eμ Systems who was using a diff amp shaper. I could see that their sine was over-rounded, and was delighted to give them the correct value. That is, to tell them something. No such luck! They knew they didn't have minimum THD, and knew that something like 79 mV was optimal. Why were they overdriving? Well, as we also noted, if you use 79 mV you get lowest THD, but the "sine" on the scope has just a noticeable trace of the peak of the driving triangle. It looks bad! Can't have that in a commercial product. To my eye, what they had was slightly too fat (driving at about 100 mV), and you could just notice the harmonic distortion. But you really don't need an audibly perfect sine wave, while an obviously visually flawed one is unacceptable. Marketing lesson. Marketing trumps engineering.]

[11] Moog, R. A., "Voltage-Controlled Electronic Music Modules," ***J. Aud. Eng. Soc***., Vol. 12, No. 3, pp 200-206, July 1965
http://www.moogmusic.com/legacy/voltage-controlled-electronic-music-modules
http://moogarchives.com/aes01.htm
Also shown in "Some Simple Good Design Ideas," Electronotes Application Note No. 200, Dec. 25, 1980 along with some historical perspective. [**Reality Note 2:** Moog recognized that this simple full-wave rectifier was imperfect and needed a small glitch-removing capacitor. Still - one transistor – hard to resist! For the most part, it was inaudible. In fact, as in the Reality Note 1 above, do we really need a sine wave at all – let alone a good one? Given that we generally want to dynamically filter a complex spectrum, we might well conclude NO.]

[12] "ENS-76 VCO Option 2", ***Electronotes*** Volume 9, Number 75 March 1977 pp10-14

[13] "Triangle-to-Sine Conversion Based on Power Series, Electronotes Application Note No. 116, Dec. 25, 1978.

[14] "Fourier Series and DFT", Electronotes Application Note No. 364 April 5, 2006 http://electronotes.netfirms.com/AN364.pdf

[15] "A Short Presentation of FFT (DFT) Interpolation" Electronotes Application Note No. 398 July 20, 2013 http://electronotes.netfirms.com/AN398.pdf

[16] B. Hutchins, "Interpolating Samples with the DFT/FFT," ***Electronotes***, Vol. 16, No. 172, July 1988 ; B. Hutchins, "FFT – Interpolation in Time and Frequency", ***Electronotes,*** Vol. 23, No. 222 June http://electronotes.netfirms.com/EN222.pdf

[17] Numerous Write-Ups on Interpolation:
"Interpolation, Decimation, and Prediction of Digital Signals", ***Electronotes*** Vol. 15, No. 164-167 Special Issue "F" July 1986 ; "Sinc Interpolation for Sample Rate Changing at Irrational Frequency Ratios", Electronotes Application Note No, *323*, July 1992 ; "Polynomial Fitting for Sample-Rate Changing at Rational and Irrational Frequency Ratios" Electronotes Application Note No. 317 January 1992 ; "Least-Square Filters for Non-Integer Delay and Interpolation", Electronotes Application Note No. 316 December 1991; "Interpolation with a Minimum-Norm Polynomial" Electronotes Application Note AN-368 Dec. 2006 http://electronotes.netfirms.com/AN368.pdf

[18] B. Hutchins, ***Musical Engineer's Handbook***, Chapter 2c, pp 8-9, or just about any book on communications theory.

[19] S. Goldman, ***Frequency Analysis, Modulation and Noise***, Dover (1967) (and others)

APPENDIX – PROGRAMS

Symmetrized Ramp Study Code

```
% en228sr
saw=[-1:.02:.98];
sq=-[ones(1,50),-ones(1,50)];
SAW=abs(fft(saw));
SQ=abs(fft(sq));
figure(1)
subplot(221)
plot([-10 110],[0 0],'c')
hold on
plot(saw,'or')
axis([-10 110 -1.2 1.2])
hold off
subplot(222)
plot([0 0],[-100 100],'c')
hold on
plot([-10 30],[0 0],'c')
plot([0:99],SAW,'or')
axis([-2 20 -5 40])
subplot(223)
plot([-10 110],[0 0],'c')
hold on
plot(sq,'or')
axis([-10 110 -1.2 1.2])
hold off
subplot(224)
plot([0 0],[-100 100],'c')
hold on
plot([-10 30],[0 0],'c')
plot([0:99],SQ,'or')
axis([-2 20 -10 80])
```

```
%
a=0
sr=saw-a*sq;
figure(2)
SR=abs(fft(sr));
subplot(421)
plot([0:99],sr,'or')
axis([-10 110 -1.2 1.2])
subplot(422)
plot([-10 30],[0 0],'c')
hold on
plot([0:99],SR,'or')
hold off
axis([-2 20 -5 40])
%
a=.25
sr=saw-a*sq;
figure(2)
SR=abs(fft(sr));
subplot(423)
plot([0:99],sr,'or')
axis([-10 110 -1.2 1.2])
subplot(424)
plot([-10 30],[0 0],'c')
hold on
plot([0:99],SR,'or')
hold off
axis([-2 20 -5 40])
%
a=.5
sr=saw-a*sq;
figure(2)
SR=abs(fft(sr));
subplot(425)
plot([0:99],sr,'or')
axis([-10 110 -1.2 1.2])
subplot(426)
plot([-10 30],[0 0],'c')
hold on
plot([0:99],SR,'or')
hold off
axis([-2 20 -5 40])
%
a=1
sr=saw-a*sq;
figure(2)
SR=abs(fft(sr));
subplot(427)
plot([0:99],sr,'or')
axis([-10 110 -1.2 1.2])
subplot(428)
plot([-10 30],[0 0],'c')
```

```
hold on
plot([0:99],SR,'or')
hold off
axis([-2 20 -5 40])
%
%
%
ms=[[-1:0.02:0.98],-[1:-0.02:-0.98]]
mssin=sin(4*pi*ms)
a=1/4 -(1/4)*mssin;
%a=sin(2*pi*a/2);
%a=0.5*(a+1);

figure(5)
plot(a,'*r')

sr=[]
   for k=1:100
      sr=[sr saw-a(k)*sq];
   end
   figure(3)
   plot(sr,'*r')
   figure(4)
   SR=abs(fft(sr));
   plot([0 0],[-500 2500],'c')
   hold on
 plot([100 100],[-500 2500],'c')
 plot([200 200],[-500 2500],'c')
 plot([300 300],[-500 2500],'c')
 plot([400 400],[-500 2500],'c')
 plot([500 500],[-500 2500],'c')
 plot([600 600],[-500 2500],'c')
 plot([700 700],[-500 2500],'c')
 plot([800 800],[-500 2500],'c')
 plot([900 900],[-500 2500],'c')
 plot([1000 1000],[-500 2500],'c')

 plot([0:9999],SR,'or')
   hold off
   axis([-25 1050 -200 2000])
```

MPWA Study Code

```
% en228d    MPWA
s=-0.975:.05:0.975;
   s=[s s]
   ss=[]
   for k=1:170
      ss=[ss s];
   end
   ss0=ss;

figure(1)
   subplot(211)
   sss=ss(1:600);
   SSS=abs(fft(sss));
   plot([0:599],SSS(1:600),'or')
   axis([-20 320 -15 210])
   sound(ss0,16000)
   pause


   m=[[1:5:36][31:-5:1]]
   m=[m m m m m m m m m m m m]

   ss=[]
   for k=1:168
      ss=[ss  s( m(k):1:m(k)+39 ) ];
   end
   sound(ss,10000)

   figure(1)
   subplot(212)
   ssss=ss(1:600);
   SSSS=abs(fft(ssss));
   plot([0:599],SSSS(1:600),'or')
   axis([-20 320 -15 110])
   sound(ss,16000)
   pause

   figure(4)
   subplot(211)
   plot([-50 750],[0 0],'b')
   hold on
   plot(ss0(1:700),'or')
   hold off
   axis([-25 700 -1.1 1.1])
   subplot(212)
   plot([-50 750],[0 0],'b')
   hold on
   plot(ss0(1:700),'c')
   plot(ss(1:700),'or')
   hold off
   axis([-25 700 -1.1 1.1])
```

```
figure(5)
subplot(211)
plot(ss0(1:1500),'r')
subplot(212)
plot(ss(1:1500),'r')


figure(6)
subplot(211)

SSS=abs(fft(sss));
plot([0:599],SSS(1:600),'or')
subplot(212)

SSSS=abs(fft(ssss));
plot([0:599],SSSS(1:600),'or')
axis([110 130 0 50])
```