Magic Sinewaves: An Executive Summary

Don Lancaster
Synergetics, Box 809, Thatcher, AZ 85552

http://www.tinaja.com   don@tinaja.com
(928) 428-4073

Digitally derived power sinewaves are crucial to solar synchronous pv inverters, industrial motor drives, power quality conditioners, and hybrid vehicles.

Major goals of such digital sinewave generation including offering the **maximum possible efficiency** by using the fewest of simplest possible switching transitions; offering the **lowest possible distortion** by zeroing out a maximum number of low harmonics that impact power quality, whine, vibration, and circulating currents; and by using **all digital techniques** that are extremely low end microprocessor and/or microcontroller friendly.

Some recent and highly unexpected solutions to a new class of mathematical functions have led to a group of **magic sinewaves** that have the remarkable property of using the **fewest** possible number of energy-robbing switching transitions to precisely **zero out** the **maximum** possible number of low order harmonics. All in an all-digital and highly microcomputer manner.

Key advantages of **magic sinewaves** include…

- **ANY chosen number of low harmonics can in theory be forced to zero. Or, under real-world quantization, can get reduced to astonishing low (-65db or less) levels.**
- **The ABSOLUTE MINIMUM number of efficiency-robbing switching transitions are needed to force the MAXIMUM number of zeroed low harmonics.**
- **Switching losses are further reduced by HALF-BRIDGE, rather than full bridge switching.**
- **Variations can provide full THREE PHASE COMPATIBILITY while still zeroing a useful number of low harmonics.**
- **Implementation is TOTALLY DIGITAL and fully compatible with economical low-end microprocessors.**
- **Modest storage needs combine with PRECISE CONTROL of both amplitude and frequency.**

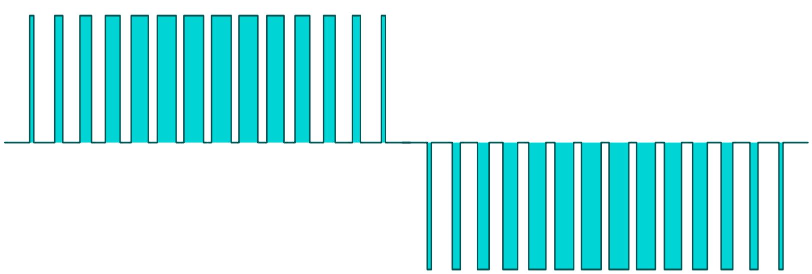
And here are the present magic sinewave limitations...

- **As with any digital sinewave generation, filtering is required to separate the sharp edge artifacts from the fundamental. Such artifacts are remarkably high in frequency in a typical magic sinewave implementation.**
- **The first two UNCONTROLLED harmonics can be quite large but NEVER exceed the fundamental amplitude.**
- **Present implementations limit magic sinewaves to power line frequencies, possibly up to 400 Hertz.**
- **While a wide frequency range can be accommodated, the response does NOT extend down to dc.**
- **Unusual programming techniques are required as each and every microprocessor clock cycle is critical. As many as 44,000 or more microprocessor instructions may be needed per power line cycle.**
- **Present implementations best separate the frequency setting from the actual magic sinewave generation.**

In some implementations, **each pulse edge zeros one odd harmonic**, and thus **guarantees** the **minimum** switching energy losses for the **maximum** number of zeroed harmonics. While many hundreds (or even many thousands) of harmonics may be zeroed, an ever increasing number of pulse edges is required to do so. With a corresponding drop in efficiency and program complexity. Evaluiation devices currently under development zero out all harmonics up to the 44th.

Magic Sinewave Appearance

Here is what a typical seven pulse per quadrant magic sinewave might look like...



This can be viewed as a highly specialized form of **PWM** pulse width modulation. One that has **far fewer** transitions than normal for significantly higher switching efficiency. And one that uses half bridge rather than full bridge switching for a further **2X** efficiency gain.

For **n** pulses per quadrant, we can see a **4n** carrier that is precisely **phase locked** to an initial **zero phase reference**. The carrier is further 100 percent modulated in that **a zero amplitude results in a zero pulse width**. As the output amplitude increases, the individual pulses also increase in an exacting and highly specific (but not quite proportional) manner.

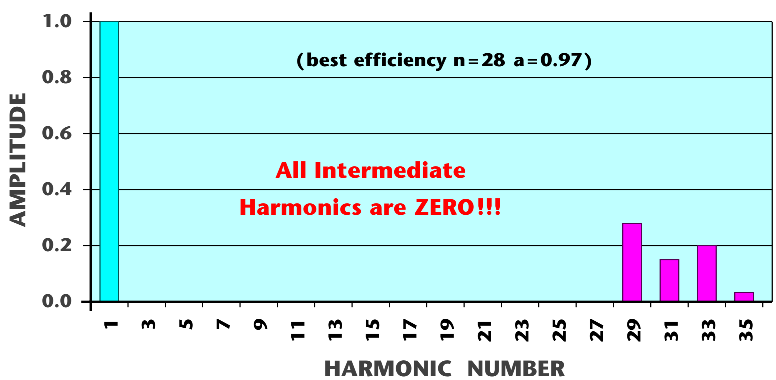
This "magic" waveform has two rather remarkable properties...

- **The first non zero harmonic is at a frequency of 4n+1. ALL INTERMEDIATE HARMONICS ARE ZERO.**
- **Most pulse edges (indirectly) zero an odd harmonic. GIVING THE HIGHEST POSSIBLE SWITCHING EFFICIENCY.**

In this waveform, there will be no even harmonics because of symmetry considerations. Most pulse edge can be thought of as **zeroing out one odd harmonic**. For seven pulses per quadrant, there are thirteen pulse edges that (working in concert) zero out harmonics 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, and 27. Plus a final pulse edge that indirectly (and again working in concert) sets the fundamental amplitude.

Real world quantizations will increase the true zero harmonics to very low values. For 8-bit compatible systems, the intermediate harmonics can normally all remain well below -65 decibels compared to the fundamental.

Here is a typical pre-quantized spectrum...



For **n** pulses per quadrant, **all intermediate harmonics up to 4n are zero**. The first two uncontrolled harmonics are fairly large and must be suitably filtered.

These first two significant harmonics range upward from approximately **one-third the fundamental** near unity amplitude. On down to nearly **matching** the amplitude height for near zero fundamental amplitudes. Unlike conventional PWM, the undesirable frequencies **never** exceed the sought after fundamental.

Proper system design will require **low pass filtering** of these higher order harmonics. Compared to other digital sinewave generation methods, these unwanted harmonics are quite high in frequency and thus should be fairly easy to deal with. As before, real world quantization will increase the zero intermediate odd harmonics to finite but acceptably low real values.

Magic Sinewave Math

An understanding of **Fourier Series** is essential to grasp the **Magic Sinewave** concept. The **classic Fourier Series** works well for analysis and avoids windowing and similar problems associated with newer FFT Fast Fourier methods. Classic analysis is acceptably fast for harmonic zeroing well into the hundreds.

Any single pulse will make this **Fourier** contribution…

cos(1*p1s) - cos(1*p1e) = ampl*pi/4

And its harmonic **j** contribution will be…

cos(j*p1s) - cos(j*p1e) = ampl*pi/4j

From which we can write the **Magic Sinewave Equations** needed for a seven pulse per quadrant, twenty eight pulse total waveform…

```
cos( 1*p1s )-cos( 1*p1e )+...+cos( 1*p7s )-cos( 1*p7e ) = ampl*pi/4
cos( 3*p1s )-cos( 3*p1e )+...+cos( 3*p7s )-cos( 3*p7e ) = 0
cos( 5*p1s )-cos( 5*p1e )+...+cos( 5*p7s )-cos( 5*p7e ) = 0
cos( 7*p1s )-cos( 7*p1e )+...+cos( 7*p7s )-cos( 7*p7e ) = 0
cos( 9*p1s )-cos( 9*p1e )+...+cos( 9*p7s )-cos( 9*p7e ) = 0
cos(11*p1s)-cos(11*p1e)+...+cos(11*p7s)-cos(11*p7e) = 0
cos(13*p1s)-cos(13*p1e)+...+cos(13*p7s)-cos(13*p7e) = 0
cos(15*p1s)-cos(15*p1e)+...+cos(15*p7s)-cos(15*p7e) = 0
cos(17*p1s)-cos(17*p1e)+...+cos(17*p7s)-cos(17*p7e) = 0
cos(19*p1s)-cos(19*p1e)+...+cos(19*p7s)-cos(19*p7e) = 0
cos(21*p1s)-cos(21*p1e)+...+cos(21*p7s)-cos(21*p7e) = 0
cos(23*p1s)-cos(23*p1e)+...+cos(23*p7s)-cos(23*p7e) = 0
cos(25*p1s)-cos(25*p1e)+...+cos(25*p7s)-cos(25*p7e) = 0
cos(27*p1s)-cos(27*p1e)+...+cos(27*p7s)-cos(27*p7e) = 0
```

At first glance, these equations seem daunting, but they really are just requesting a desired fundamental combined with forced zeroing of the first 28 harmonics.

Until some recent considerable serendipity requiring exceptionally tedious brute force methods (involving over a decade of research), there was not the slightest hint that **any** solutions at all existed for the above equations. Let alone any ones highly useful for both power quality and energy efficiency. Hence the believed uniqueness and originality of the **Magic Sinewave** approach.

I presently believe that **no closed form method of solving the above equations is currently known**. At present, an **iterative** JavaScript **custom tool** that is related to **Newton's Method** can be effectively used. One that is quite fast, highly accurate, and rapidly converging.

Solution proceeds by **making a good guess** as to the desired result. An error function is derived using a **trig identity** and then used to improve the guess via Newton's Method and **Gauss-Jordan Reduction**.

The process is typically repeated five times to provide results beyond ten decimal place accuracy. The harmonic zeros and the amplitude simultaneously converge.

The present calculators are useful for **15** or fewer pulses per quadrant, reporting harmonics as high as the **113th**. They cover **Delta Friendly**, **Best Efficiency**, and **Bridged Best Efficiency** magic sinewave types.

Extensions, explorations, more specialized calculators, and other results are available on a **custom consulting** basis.

The above equations can be shown to be related to **Chebycheff Polynomials**. Further analysis of which leads to some profoundly "bare metal" power equations. But, sadly, does not seem to simplify the iterative calculations needed for solution.

These fundamental power equations very strongly suggest that...

> **No simple closed form solution to Magic Sinewave equations is likely to exist.**
>
> **No more efficient solution is likely to exist for a given number of zeroed low harmonics.**

Other sets of **Magic Sinewave** equations lead to different solutions. Our above equations are an example of **Best Efficiency** magic sinewaves.

Another solution set of interest leads to some **three phase compatible** or "**delta friendly**" magic sinewaves. These meet some exacting needs of three phase power systems while still zeroing out a respectable number of low harmonics. These delta friendly versions also have advantages of requiring less memory storage and calculating significantly faster.

A group of highly sophisticated **JavaScript** Magic Sinewave calculators are now available and have been posted to **http://www.tinaja.com/magsn01.asp** With solutions presently offered as high as **384** zeroed low harmonics.

A solution set for the above equations would be...

Pulse Positions in Degrees		Target Amplitude:	0.970000000		
		Target Power:	0.9409		
p1s:	10.24045701482	Fundamental Amplitude:	0.97000000		
p1e:	12.37453448393	THD 2-28 is	0.0000000000 percent.		
p2s:	20.53940225855	h03:	0.00000000000	h27:	-0.00000000000
p2e:	24.75285470128	h05:	0.00000000000	h29:	-0.28097991231
p3s:	30.95837848563	h07:	-0.00000000000	h31:	-0.15202977074
p3e:	37.14383081227	h09:	0.00000000000	h33:	0.20474366382
p4s:	41.56706542190	h11:	-0.00000000000	h35:	0.17750740350
p4e:	49.57368363981	h13:	0.00000000000	h37:	0.04412815293
p5s:	52.45588082432	h15:	0.00000000000	h39:	0.00660309324
p5e:	62.12795008492	h17:	0.00000000000	h41:	-0.00078945970
p6s:	63.77803848365	h19:	-0.00000000000	h43:	0.00211106247
p6e:	75.13315211929	h21:	-0.00000000000	h45:	-0.00342276315
p7s:	75.93480957382	h23:	-0.00000000000	h47:	0.00562585723
p7e:	89.76625289333	h25:	-0.00000000000	h49:	-0.00924395073

A **recent version** of the calculators provides solutions well beyond ten decimal place accuracy and harmonic analysis well into the hundreds. And there is every reason to believe that these results do in fact approach true harmonic zeros. That represent actual true solutions to the magic sinewave equations.

Real-World Quantization

It is important to note that **extreme precision is required in both the analysis and generation of magic sinewaves**.

This problem is severely compounded when translated to many real-world devices using 8-bit microprocessors. Primarily due to **quantization** or roundoff errors. Coding can become extremely tricky at "pinch points" in the timing.

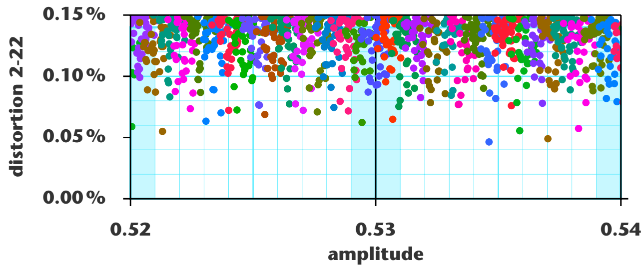
Time delays and pulse widths accurate to around **twelve bits** appear to be needed to guarantee "zero" harmonic rejection values that are **-65 decibels** or more below the fundamental. Some sneaky tricks are required to produce this precision level in an 8-bit microprocessor environment.

For instance, it is possible to split a time delay into **two** elements. A **calculated** component that is proportional to amplitude and a **residue** component that can be found through **table lookup**. Depending upon specific need, the calculated component might be directly or inversely proportional to amplitude and may or may not be threshold truncated. The net result is one method of obtaining 12-bit precision in an 8-bit environment.

The time resolution of the microcomputer also causes quantizations. As a result, **very high clocking frequencies will usually be involved**. A **60 Hertz** power sinewave might require an instruction rate of **2.5 megahertz** and something around **42,000 executed instructions per power line cycle**. In the case of a **PIC** or similar microprocessor with a **4X** clock, **about a ten megahertz clock frequency may be needed for a 60 Hertz sinewave**. For this reason, **magic sinewaves are often limited to power line and low audio frequencies**.

A further quantization improvement in precision can result from selecting some **nearby** magic sinewave, rather than a supposedly optimal one. This is possible because quantization tends to produce a **dripping stalactite** type of result. By investigating a few hundred thousand or more nearby magic sinewaves per amplitude, one or more results can typically be found that are often **two to ten decibels** lower in harmonic distortion than the initial cut.

Here is a "group portrait" that shows this effect...



Firmware Considerations

While low end applications pretty much demand a **PIC** or similar microprocessor, use of wider bus devices or even **DSP** digital signal processing techniques may prove of advantage for higher end development.

Because of the exacting requirements of **Magic Sinewaves**, some ultra precise and non-mainstream programming techniques will usually prove of value. Timing and precision requirements can often end up unusually strict.

Before firmware development can even begin, four time intensive tasks are required. First, each and every desired amplitude needs carefully analyzed using the **JavaScript Calculators** and generating a **list** of pulse positions and widths. Suitable calculators are available at **http://www.tinaja.com/magsn01.asp**.

Second, several tens of thousands to several hundred thousand "nearby" solutions should be explored by **intentional jitter techniques**. This will typically result in candidate amplitudes that have reduced distortion of **two to ten decibels**. It is very important to **keep frequency and amplitude independent** during the exploration of alternatives.

Third, the frequency and amplitude values should be sent to a **Fourier Analysis** software simulation program that can verify that each and every amplitude does in fact offer acceptable low intermediate harmonics. Finally, the verified values need converted into combinations of calculated ramps and residue delay values.

Only after these tasks are complete can the actual programming begin. Firmware guidelines have been published as **http://www.tinaja.com/glib/mspicpro.pdf**.

Generation of magic sinewaves seems simple enough as all that is involved is repeatedly outputting port patterns followed by precision delays. Our first guideline is by far the most crucial...

> **Magic Sinewave timing must be exceptionally precise and perfectly equalized for useful low harmonic rejection!**

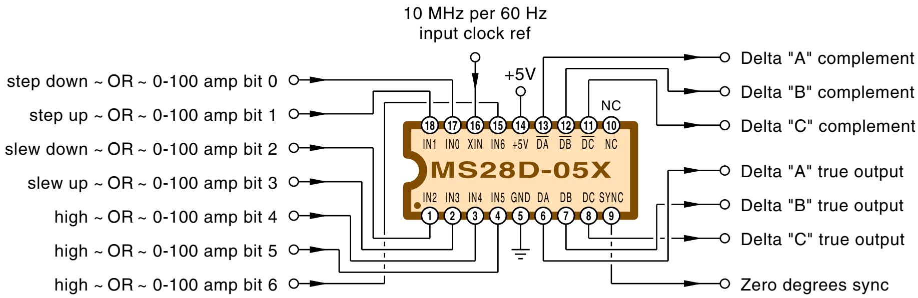
Each and every path through the code **must take precisely the specified amount of time**. No more, no less. Thus, some highly unusual programming skills and techniques will almost certainly be required.

For instance, **it appears best to outsource the actual frequency generation**, by varying the microprocessor clock at some high multiple of the desired power line output frequency. Use of "bare metal" machine language programming is almost certainly a must. Compiling from a higher level language is virtually certain to cause timing and jitter problems.

As we have already seen, there are various tricks that can be used to allow an 8-bit microprocessor to provide 12-bit accuracy. Some of which involve splitting time delays into two or more tasks. The luxury of extensive subroutines may not be permitted at code "pinch points". Thus linear coding will often be the most appropriate solution.

Time shifting or "pipelining" of certain routines may be required to fit the available intervals. As might precaching of certain data values.

Here is an example of an earlier three phase magic sinewave generator...



This was an earlier prototype delta friendly design. It has been simulation verified and is presently still undergoing full speed bench verification tests and real-world harmonic analysis.

The thinking was to provide a **dual mode** input capability. There are seven binary input lines. Inputs that are coded **0** through **100** will immediately cause the chip to output that amplitude. By holding the three most significant input bits high, the device converts into four pushbutton operation of **step down**, **step up**, **slew down**, and **slew up**.

While this design easily fits a 2K memory space, use of 4K or 8K devices are more appropriate for higher harmonic rejection needs. In general, a best efficiency magic sinewave needs **2n** 8-bit values stored per amplitude, while a delta friendly needs only **n**. Where **n** is the **number of pulses per quadrant**. The majority of space in typical magic sinewave firmware is taken up by the amplitude table lookups.

While a hundred linear spaced amplitudes (plus zero) are the norm, any reasonable number of amplitudes can be provided. Amplitudes can also be constant power, nonlinear for load compensation and equalization, or even random for candle and flame effects.

Magic sinewaves are also possible on extremely small microcontrollers provided only a very few amplitudes are required.

For Additional Assistance

Visit the many **Magic Sinewave** files at **http://www.tinaja.com/magsn01.asp**. Or else email **don@tinaja.com**. Or call **(928) 428-4073**.