The Saga of the Dripping Stalactite

Don Lancaster
Synergetics, Box 809, Thatcher, AZ 85552

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

Some **Applied Mathematics** in the form of **playing with long strings of funny numbers** can end up anything from an obsessive fascination to a stunning **energy efficiency breakthrough**.

Let us first gather together several of our more obscure **Guru's Lair** "fun with numbers" resources. Followed up by the discovery timeline on how a two decade long obsessive fascination with applied math led to the **Magic Sinewave** energy efficiency breakthrough…

Barker Codes & Correllation— These codes give a strong value when multiplied by themselves in phase and a weak value when multiplied by themselves when timeshifted or against external noise. Which becomes super important when you are extracting signals from deep noise, for **GPS**, for spread spectrum, or when otherwise having several signals simultaneously communicating interference free over the same channel. (Go to **Hardware Hacker 54** in the linked archive.)

Digital Filters— Digital filters can do things that analog cannot because they can look both **backward** and **forward** through time. They are often rather simple to implement, consisting only of delay, scaling, and summing. The only little trick is picking the right scaling **coefficients** ahead of time to get your desired results. Also see **MUSE107.PDF** for a more detailed design example.

Equally Tempered Music— There is only **one** magic combination of 8-bit data values that can give you the equally tempered music scale without generating at least one sour note. This is the magic sequence **116-123-130-138-146-155-164-174-184-195-207-219-232**. Also see my **CMOS Cookbook** for further details.

Fractal Ferns— These fantastic self-replicating, self-repeating, and self-scaling patterns assert themselves often in nature. With coastlines and mountains being typical. Curiously, **the shorter the ruler you use to measure a coastline, the longer it gets!** As per Mandelbrot and company. And **the fern** is by far the most simple and spectacular demo. Also see our **PostScript** library. Especially our fern **Fractal Image Converter**.

Pseudorandom Number Generators— Otherwise known as **noise that repeats**, these are made from long shift registers with carefully selected feedback taps. Any short sequence of output numbers appears to behave pretty much like truly random output. (See **Ask the Guru 1** in the linked archive.) Lots more in my **Apple Assembly Cookbook** and my **CMOS Cookbook**.

Cubic Splines— These efficiently generate smooth continuous curves from very sparse control data points. They get used for everything from **PostScript Fonts** to computer graphics **to machine tool cutting paths**. One popular form is called a **Bezier** curve. An **Intro Tutorial** is found here.

Active Filters— By making a subtle change in the math of classic active filters, new "equal component value" designs evolve that are much easier to use and separate their frequency and damping determining elements. These use opamps to replace inductors and are extremely versatile.

n-connectedness— Some simple yet subtle math shows you **how to connect everything to everything else**. And has extremely diverse uses from letting a microcomputer port drive a surprisingly high number of different LEDS to unusual "string art" to web usage patterns to explaining why organizations become unwieldy even if only a few new members are added.

Binary Chain Codes— These unusual sequences have the unique property that a short sample of them reveals that sample's **position** in the entire code. Leading to all sorts of robotic and automation absolute position sensing apps. Chain codes often close on themselves for continuous sequencing. Only an **n** bit sample is needed to find where you are in a code of length 2^n.

Wavelets— Wavelets are an exceptionally powerful new math tool. One causing revolutionary developments in most everything from cardiology through seismology. But most significantly to new methods of video and still picture compression. Wavelets can uniquely supply both the big picture and fine detail at the same time. Here's an **intro**.

Basis Functions— Also known as **Bernstein Polynomials** or **high resolution cubic splines**, these are the secret to changing the size or distorting an image. And doing so without artifacts and actually **improving** the final appearance. Extensive use examples **here** and **here**.

Binomial Coefficients— These see lots of statistical use such as "How many combinations of six coins are there that will have two heads and four tails?" And sure got important in my earlier **Magic Sinewave** work where the question in disguise was "How many six bit words are there with only two ones in them? Per this **example**.

Breshenham's Algorithm— Converting from vectors to steps is important for everything from rasterizing computer graphics to **generating machine tool paths**. One of the most fundamental graphic routines.

Shuffling right along— There's a subtle gotcha in the "obvious" shuffling algorithm that introduces bias errors. Here's the correct way to, say, shuffle a deck of cards or **rotate the position** of **advertising banners**. More sourcecode **here**.

Nonlinear Graphic Transformations— Remapping of ordinary graphics onto unusual shapes has all sorts of interesting uses. The real biggie here is perspective, but other examples include spherical, cylindrical, rootbeer, scribble, isometric, glyph path, and star wars transforms. Lots of additional examples **here**.

Chebycheff Polynomials— These are simply a fancy way to manipulate trig multi angle identities. As such, they see all sorts of interesting uses in everything from my **Magic Sinewves** (see below) to **Active Filters**. They are surprisingly simple to generate, and often produce the "best" of all possible solutions.

Image Correction— By selectively lengthening or shortening a **bitmap** pixel line, **image distortion** can be either eliminated or enhanced. Similarly, by **comparing individual bitmap pixels** against a **mask**, adjustments can be made in brightness, contrast, gamma, hue, knockout, or **transparency**.

Fibonacci's Sunflowers— The numeric sequence **1-2-3-5-8-13-21-34-55-89...** occurs repeatedly in nature, since things tend to expand or contract in proportion to their present size. Tutorial and sourcecode shows how to draw sunflower-like patterns in **PostScript** using an incredibly magic angle of **138.58776** degrees.

Gaussian Elimination— This is a standard method to solve **n** linear equations in **n** unknowns, and is based on zeroing certain matrix values and making others unity. Such rearrangements ultimately let the answers pop out by inspection. It largely replaces older **Determinant** methods.

Fourier Series— Relating time to frequency is crucial for advanced engineering analysis. The most profound and fundamental tool for this is the **classic Fourier Series** and its newer **Discrete Fourier Transform** and **Fast Fourier Transforms**. Extensively used for the **analysis** of our **Magic Sinewaves** discussed below.

RMS Power Calculations— Until recently, low cost methods of measuring true electronic power of unusual waveforms did not exist at all. Which led to all sorts of "**not even wrong**" claims about circuit **energy efficiency** and "**overunity**" hogwash. Additional info is also found **here** and consulting services **here**.

Heap Sorting— The heapsort data structure comes remarkable close to looking exactly like a pile of cowshit. But this arcane and hard to understand algorithm sorts high quantity items much faster than **conventional approaches**. Growing at **n log (n)** rather than $(n)^2$.

Taylor Series— This math tool is most useful to create or eliminate nonlinearities. It approximates a real-world response with a level, a slope, a parabola, a cubic, and as many additional terms as are needed. Also useful to **draw a curve through n data points**.

The Magic Sinewave Timeline

My **Magic Sinewave** story starts in 1981 with a 6502 programming class I was teaching at **EAC**. I had encouraged several students to try and feed long binary sequences to an **Apple IIe** speaker to see if they couldn't get **anything interesting** out of the two-state speaker in the way of nice sounding timbers or even chords.

Every once in a while, a fairly pure low tone would come out that only had minor buzz or whine on top of it. Which got me to thinking about whether conventional digital sinewave synthesis could somehow be improved.

I felt that an ideal digital sinewave synthesizer for power electronics should…

- **Offer lots of precisely controllable amplitudes.**
- **Be totally digital and low end microcontroller friendly.**
- **Completely zero out as many low harmonics as desired.**
- **Use the fewest switching events for highest efficiency.**
- **Have no harmonics stronger than the fundamental.**
- **Include an optional three phase compatibility.**

Such a selection of **Magic Sinewaves** would be enormously useful to dramatically improve the **energy efficiency** of ac induction motors, electric vehicles, pv solar panels, power quality conditioning, brushless servo motors, telecomm, battery inverters, and aerospace power. As per **this tutorial**.

I first started off by "brute force" investigating **all of the possible** shorter binary sequences. Not surprisingly, things got rather ugly beyond **n=32** with its **over four billion** states to be explored. Shorter sequences did give an occasionally useful amplitude with somewhat decent close in harmonics. But these were very few and far between.

Ferinstance, there are only **nine** useful 30-bit words of differing amplitudes having no third or fifth harmonic. From over a billion possible total words. All of which strongly suggested that **very long** binary sequences would be needed if anything new was going to be discovered. A choice of a 420 bit word seemed to lead to several interesting possibilities. By using quarter wave symmetry, we'd only have to work with a mere 105 bits per quarter.

And 105 happens to be the product of **3**, **5**, and **7**. Which suggests that we should be able to find lots of solutions having none of these harmonics present. Nor any of their 9, 15, 21… product harmonics. We can write 35 bit equations that force the third harmonic to zero. And 21 bit equations that force the fifth to zero. And 15 bit equations that force our seventh to zero.

These 71 equations force certain bits to be **dependent** on one another. Which means we can get down around **105-71=34** independent bits that we'll have to exhaustively search.

Fortunately, there are some other tricks we can pull. We are dealing with **binary** numbers here. If **k = w + x + y + z** and w through z are in binary, then k can only assume **sixteen** different values. Further, if k is also binary, it can only assume the **five** values of **0+0+0+0, 0+0+0+1, 0+0+1+0, 0+1+0+0** or **1+0+0+0**.

All of which fairly quickly gets you down to only a few thousand sequences to explore. I picked out the best hundred of these and **published them all** in **Circuit Cellar** back in early 1997.

While these sequences sure seemed interesting at the time, they remained far from optimal. Because of their still having fairly high distortion, way too many efficiency-robbing switching transitions, high amplitude jitter, several missing values, strong filtering requirements, and fairly high storage needs.

The Dripping Stalactite

So, I doggedly continued exhaustively exploring higher bit lengths. 768 bits seemed to have some advantages. But it still gave only erratic and mediocre results at best. At that point, I decided to focus on **contiguous** groups of ones that gave **fewer** pulses and thus higher efficiency. This also vastly reduced the patterns to be explored.

Why higher efficiency? Because the fewer the switching events, the fewer the dynamic switching losses. These dynamic losses typically account for around **one-half** of the total controller losses in most power electronics systems.

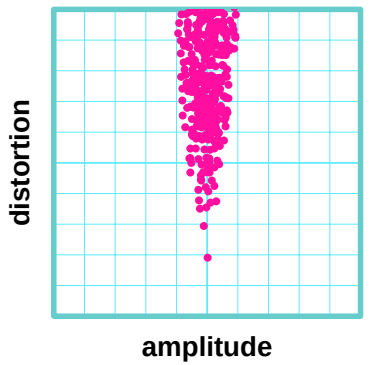
I also created a mythical **minimum visual pollution theorem** guideline. One that said that if a pulse pattern didn't "look" nice and sinewaveish, then it probably wasn't worth bothering over. Thus, you reasonably would expect pulses to get consistently **wider** as they approached 90 degrees of a sinewave quadrant. And you would expect them to be pretty much evenly spaced as well.

This theorem ended up getting slightly violated later on with our **delta friendly** magic sinewaves, but at this point it was a useful guideline. And again very much reduced the patterns to be tested.

By working in **quadrants**, all even harmonics and any dc term could automatically be eliminated. The number of states to be explored got further reduced. A full sinewave thus is created by **mirroring** the quadrant for the first half cycle, and both **mirroring** and **flipping** for the second.

Working by quadrants also greatly reduces the storage needs for each amplitude value. Ultimately, as few as five to eight 8-bit stored values were required for each amplitude. Thus easily fitting inside a smaller microcomputer.

As the various bit patterns were tried, a consistent data structure kept showing up. One that looked exactly like a **dripping stalactite**…



Each dot represented the distortion and amplitude gotten from a single trial of selected pulse positions. At the time, I did not have the faintest clue **why** such an unusual shape was evolving. But, as a **caver**, I knew a lot about stalactites.

Much later it became obvious that we were looking at necessary **quantization errors**. Caused by whole integer approximations to needed exact values.

At any rate, two features of the stalactites became apparent: There were often **drips** at the bottom that were consistently a lot better than the rest of the gang.

"Better" being lower distortion and closer to the target amplitude. Repeated careful testing revealed that, yes, these were real. And, the missing **tip** of the stalactite seemed to "point" to a yet-to-be-discovered super duper **zero** distortion result. Our holy grail.

It seemed that if any **magic solutions** were to ultimately be found, they would involve **extremely long** binary words. Our current production **evaluation chips** typically use binary words of **41,664** bits and even higher! Clearly, the "filtered exhaustive search" method that got us this far wasn't going to hack it.

So, I tried a different tack and temporarily "went analog". I grabbed one of the most promising drips and then tried to optimize it. At the time, I was hoping for nothing more than a decibel or two of improvement at best. I went with what I named my "**shake the box**" scheme.

In which I would take a pulse edge and make a **one-tenth** degree change in its position, seeking an optimum. I'd then repeat for the other pulses. Followed by a **one-hundredth** degree change and then a **one-thousandth** degree change. I'd then continue till things did not get any better.

This approach turned out to be identical to a standard math "tough equation" solving process called **Newton's Method**…

To solve a nasty math problem by Newton's Method, first guess at a good answer. Calculate that guess.

Then make a small change in one of the variables to improve your results. Continue for all variables.

Repeat with progressively smaller changes until you get acceptably close enough to a true solution.

Lo and behold, doing so with the drips quickly lead to…

The 3:17 am Ephitany

Not only was the drip distortion reduced, but it disappeared entirely! Starting with seven pulses per quadrant, I ended up with astonishing low distortion levels for the first **twenty-six** harmonics! **The holy grail was found!**

Or was it?

Any time you find a "too good to be true" result, STOP IMMEDIATELY and TRY TO PROVE YOURSELF WRONG!

I sent the result to another mathematician who agreed that this, against all odds, was real. So, it was off to the races. How many **Magic Sinewaves** were there? Were there better ones? Had we already achieved the best possible efficiency?

It seemed way too early for any solid theory, so I started a brute force exploration. Similar to hunting for …

Canyons in the Bajada

I tried to visualize the **Magic Sinewave** space as terrain common to the Basin and Range desert Southwest. In which you had an upland sloping **bajada** that was so heavily brush covered so you should not see very far.

In the bajada were some deep canyons, and at the bottom of the deep canyons, streams representing a continuum of Magic Sinewave solutions. You would take a guess where to start and wander around till you found a canyon. Then you would descend to stream level. Finally, you would work up and down the stream, while extracting magic sinewaves of sequential amplitude values.

The question was how many canyons were there and which were the very best? By taking our one drip solution and then simply fattening or shrinking the pulses slightly, a series of Magic Sinewaves of amplitudes 0 to 100 was quickly found. I ended up calling these the **normal** series.

I next tried to improve my exploration tools. I was up against the 32 bit math limit of the **PostScript** analysis utilities I was using, so I created some **JavaScript calculators** that were both more interactive and had 64 bit math precision.

This additional eight or so decimal places of "daylight" reduced the "zero" values so low that only the fussiest of mathematicians could object that these were not "proven" true zero solutions. An engineer, of course, would shrug this off as "What else could they be?".

Moving the pulse guesses towards 90 degrees found a whole new "canyon" and a new class of magic sinewaves with an even more amazing property: **Two more harmonics were zeroed!** Seven pulses per quadrant zeroed out the first **twenty eight** harmonics! I called these the **best efficiency** series and strongly believe that **none more efficient exist or are possible**.

With "efficiency" in this case being defined as **the maximum number of zeroed harmonics for the minimum number of pulse edges.**

This once again seemed "too good to be true". Because a sampled data system of **n** events per cycle tends to have strong odd harmonics at **n-1** and **n+1**. But the mystery was resolved with some smoke and mirrors: Two "invisible" pulses of zero width and **zero energy** were really present at 0 and 180 degrees. You actually had 7-1/2 pulses per quadrant and 30 pulses per cycle total. But could pretend two of the pulses simply weren't there!

"Best Efficiency" Properties

While I still didn't yet have the exact math for the best efficiency magic sinewaves of **n** pulses per quadrant, their properties could easily be stated…

The waveform appears somewhat similar to a 100 percent pulsewidth modulated carrier of frequency 4n.

The first 4n harmonics are ZERO with a true solution and very low when quantized to properly chosen integers.

The first two uncontrolled harmonics at 4n+1 and 4n+3 are large but never larger than the fundamental.

Higher harmonics are quite low in energy and very high in frequency. Spectrum is thus spread out for separation.

Each and every pulse edge performs ONE useful task, thus guaranteeing the highest possible efficiency.

A intro summary of additional properties is **found here**. A tutorial of these findings also **appears here**. Along with evaluation chips **here**.

Other **Magic Sinewaves** were found and explored. By sharing one wider pulse at 90 and 270 degrees, you could create a class of **bridged** magic sinewaves.

More important were **Delta Friendly** magic sinewaves that were fully three phase compatible. Three phase power requires that all triad harmonics be zero and that only three half bridge drivers be used if rewiring is to be avoided.

Special and rather arcane tricks are needed to lock **pairs** of pulse edges together. The bottom line is that **highly useful three phase solutions exist.** They only zero out **3n/4** harmonics compared to the **4n** of a best efficiency solution. But they analyze **much** faster and need only **one half** of the storage words per amplitude.

The early **JavaScript calculators** tended to be rather slow on longer sequences, so they were modified to calculate only **differential changes** rather than doing a full **Fourier Series** for each and every box shaking.

This resulted in a dramatic speedup. Solutions to 384 pulses have been posted to **my website**. Counts beyond 1000 pulses per cycle seem possible. Ultra long sequences might ease filtering and audio whine at the cost of extra switching losses and extra storage. Yet still be significantly more efficient than classic PWM.

Rewriting the calculators for each new length got to be painful, so I wrote a new **PostScript program that generates JavaScript programs!** More details on this per our **consulting services**.

At this point, it was time to start thinking about switching from analysis to synthesis. And at long last looking into the actual…

Magic Sinewave Math Equations

Equations for a seven pulse per quadrant best efficiency magic sinewave are…

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

Our first equation sets the fundamental amplitude, while the rest of the equations zero out the total harmonic distortion through the first **4n** harmonics. My "shake the box" or **Newton's Method** works just fine for an iterative solution.

One tiny detail: You may get a slightly wrong amplitude on your first pass. Simply repeat with an adjusted amplitude request. For instance, if you want **0.400** and get **0.396**, try again asking for **0.404**. The process usually will rapidly converge.

We see that we have fourteen equations in fourteen unknowns. Thus…

> **Each pulse edge (indirectly) performs ONE useful task.**
>
> **Maximum harmonics are zeroed by minimum pulse edges.**

You could think of one pulse edge variable as (indirectly) setting the amplitude and the other thirteen pulse edge variables as zeroing out all the odd harmonics **3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25,** and **27**. And all the even harmonics are eliminated by way of quadrant symmetry. Apparently giving us the **best possible** energy efficiency solution to low harmonic elimination.

We might ask an obvious question here: Wouldn't it have been simpler and two decades faster to just write and solve these equations in the first place?

Well, applied math sometimes does not work out that way. There was not the least reason to suspect that **any** solutions existed at all to the above equations. Let alone ones that would lead to a major breakthrough in **energy efficiency**.

Digging Deeper

It can be useful to try and **transform** our above equations into other forms. This might give us insight into what is really coming down, might get us a step closer to actually proving that no more efficient **Magic Sinewave** solutions exist, or may let us try to solve our equations using some fast non-iterative method.

We seem to have fourteen nonlinear trigonometric multiple angle equations in fourteen unknowns. The ugliest feature of this nasty mess is obviously the multiple angle part. Any trig book should give us a few **identities** such as…

$$\cos(3\theta) = 4\cos(\theta)^3 - 3\cos(\theta)$$

$$\cos(5\theta) = 16\cos(\theta)^5 - 20\cos(\theta)^3 + 5\cos(\theta)$$

These particular funny equations can be related to the **first kind Chebycheff Polynomials**. They get rid of the multiple trig angles by replacing them with fundamental angle equations. More detail on the process **appears here**.

Chebycheff Polynomials are widely used to solve all sorts of sticky problems, especially with **Active Filters**. They share this interesting property…

> **When Chebycheff Polynomials are a good solution to a problem, they often can be proven the BEST POSSIBLE.**
>
> **Thus driving the Cheby to the Leby.**

Once you are down to the single angle equations, you can then use **x = cosθ** to **eliminate all the trig entirely!** And doing so does lead to some profoundly "bare metal" power equations. Per **these details**.

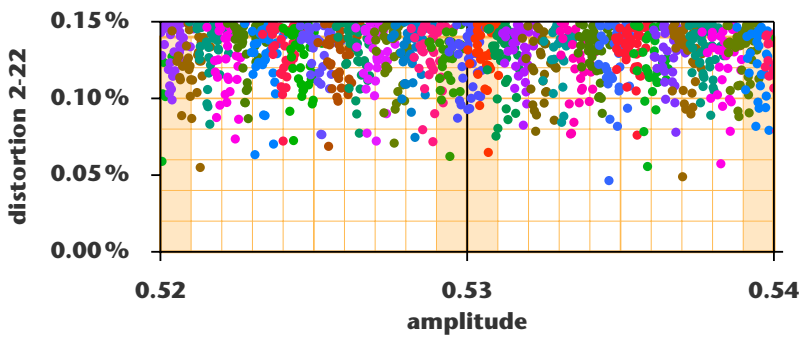
Unfortunately, the "bare metal" power equations seem to be even harder to solve than our above trig equations. Despite their **elegant simplicity**.

The odds seem very high that a best efficiency **Magic Sinewave** is in fact the most efficient at maximizing zeroed harmonics for minimum pulse edges. And that a simple "closed form" direct solution to the underlying math is very unlikely.

Tools to analyze Magic Sinewave spectra can be **found here**. Along with a full development proposal **here**. And evaluation chips **here**.

The Whole Gang

To wrap things up, let's see if we can't coax a color coded bunch of dripping stalactites to pose for a group portrait…



The number of stalactites is set by the **fastest changing** variable. In this **Delta Friendly** example, pulse edges **P2S - P1E** will have the fastest rate of change.

The **rate of variable change** (and your best achievable distortions) in turn will be decided by how finely you are **quantizing** as set by your clock cycles to output cycles ratio. As you can see, the stalactites will combine into a selectable continuum of "good" and "best" values.

In this particular **three phase** example, the value chosen for amplitude 0.53 gave a total **unfiltered** 2-22 harmonic distortion of just over **0.06** percent. Harmonics 2 through 22 were all a minimum of -65 decibels down from the fundamental. Most were **much** lower. Per this **Analysis Tool**.

Those mysterious "drips" are simply the best available quantized approximation to a perfect **Magic Sinewave** solution! For the highest possible **Energy Efficiency**.

For More Help

Additional "fun with numbers" applied math resources appear in our **Math Stuff** and **Tech Musings** libraries. **Custom Assistance** is also available.

Our **Magic Sinewave** library holds bunches of additional **energy efficiency** breakthrough support. Including these two intros **here** and **here**, a development proposal **here**, a tutorial **here**, visualizations **here**, jitter and distortion analysis **here**, lots of calculators **here**, seminars & workshops **here**, analysis tools **here**, and our latest release of evaluation chips **here**.

The **MS28D-04X** magic sinewave chips are newly available at $19.63 each plus shipping. Sourcecode and one hour of consulting is separately available for $89 additional.

You can order your samples and sourcecode **here**. They should also be shortly available on **eBay**.

Licensing arrangements for your own chip production using our sourcecode or any of its derivatives or variants are available and are quite reasonably priced. You can **email me** for further details.

Additional **Magic Sinewave** services, programming, seminars, training and project development are available **here** and **here**.

Further **GuruGrams** columns await your ongoing support as a **Synergetics Partner**.