Hardware Hacker

Using cubic splines
Switchmode resources
Electrorheological fluids
National's simple switcher
Non-ionizing radiation safety

April, 1993

Let's start with some updates to our earlier columns. Yet another approach to the $5 Navicube is a plain old low cost mechanical gyro. Which is the route that *Gyration* has chosen with their unique *GyroPoint* system. Their entire gyro is the size of a film can. It has been designed from the ground up for consumer apps.

Prototype costs, of course, are still totally outrageous. To me, spinning wheels do seem a hopelessly outdated interim solution at best. There's no doubt that micromachined silicon is ultimately the only way to go.

The CEO of a traditional gyro firm has assured me that it was "absolutely impossible" to create a $5 rate gyro. Well, there are flat out far too many emerging applications that *demand* a $5 gyro replacement. The paradigm *has* shifted, and the time is long past due for stunning cost reductions here. The choice a traditional gyro CEO has is simple: Bring out this inevitable product by yourself.

Or else go down in flames.

Here are two more Asian electronic resources: There's now an *EDN Asia* version of this popular trade journal. The *Hong Kong Trade Development Council* has a toll-free number which lists 52,000 manufacturers, importers, and exporters. Some other assistance is also available.

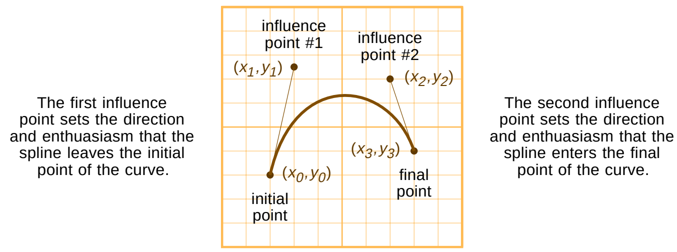
Cubic splines

There are lots of times and places where you would like a machine or a computer to create a graceful curve. Perhaps with a CAD/CAM mill, on an engineering graph, in a 3-D surface modeler, vinyl signcutter, or for any other typography, plain or fancy.

I have recently been exploring a really great scheme to elegantly deal with graceful curves. These are called *cubic splines*. They sometimes do get called *Bézier* curves, after the French mathematician who explored them.

What do we ask of a curve drawing method? First and foremost, it should look good. Second, we want it to be largely *device independent*, working

(A) Here is a cubic spline shown in its *graph space*...



(B) Here is how a cubic spline appears in its *equation space*...

$$x = At^3 + Bt^2 + Ct + D$$
$$y = Et^3 + Ft^2 + Gt + H$$

t (or time) always goes from zero at the initial point to one at the final point

(C) A faster "cube-free" form of the *equation space* math...

$$x = (((At) + B)t + C)t + D$$
$$y = (((Et) + F)t + G)t + H$$

(D) How to get from graph space to equation space...

$A = x_3 - 3x_2 + 3x_1 - x_0$	$E = y_3 - 3y_2 + 3y_1 - y_0$
$B = 3x_2 - 6x_1 + 3x_0$	$F = 3y_2 - 6y_1 + 3y_0$
$C = 3x_1 - 3x_0$	$G = 3y_1 - 3y_0$
$D = x_0$	$H = y_0$

(E) How to get from equation space to graph space...

$x_0 = D$	$y_0 = H$
$x_1 = D + C/3$	$y_1 = H + G/3$
$x_2 = D + 2C/3 + B/3$	$y_2 = Y + 2G/3 + F/3$
$x_3 = D + C + B + A$	$y_3 = H + G + F + E$

(F) Some cubic spline tools available on *GEnie PSRT*...

#606	GNZOTRX2.GPS	New TEX-like Guru Gonzo utilities
#605	BEZIER1.PS	Distance from point to Bezier curve
#590	BEZIER.PS	Bezier curve fast evaluation.
#588	FUZZYBEZ.PS	Bezier curve from fuzzy data (!)
#530	SPHERE.PS	PS spherical nonlinear mapper
#214	GURU68.TXT	Bezier curvetracing fundamentals
#205	NUBANNER.FNT	Fast & accurate parade banner freefont
#202	TWIXTBEZ.FNT	Flagwaver freefont & Bezier ct utility
#161	SLEEZOID.PS	Avuncular sleezoid utility
#190	BEZLNGTH.PS	Bezier curve length utility
#077	MEOWWRRR.PS	PostScript puss de resistance

Fig. 1 – CUBIC SPLINES are a useful and convient way of handling smooth and graceful curves by using a sparse data set. Here are several of the fundamentals you will need to start exploring these on your own.

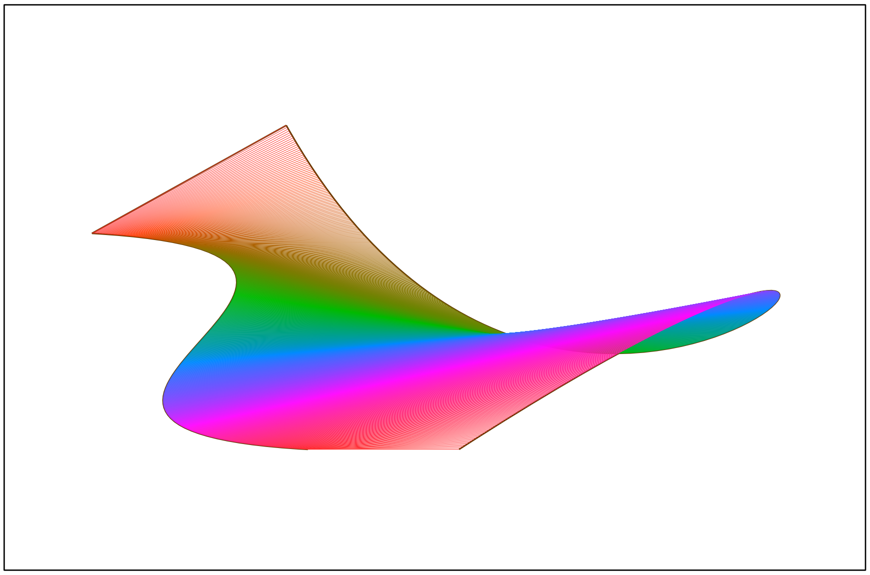


Fig. 2 – THIS AVUNCULAR SLEEZOID is a just-for-fun example of how you can stop anywhere along a cubic spline. Note how the "t" parameter changes "faster" along the "more bent" portions of the curve.

the exact same way for video screens, laser printers, typesetters, embroidery machines, or whatever. All done from the *same* ordinary textfile.

Third, we'll want it to use *sparse data*, easily described by using only a few size-independent values. Fourth, we'll want to be able to splice simple splines together into more complex curves without serious glitches.

Fifth and finally, we want it to be reasonably fast and easy to compute. And easy to understand.

The simplest way to build a curve is the *step method*. Where you move over a click and up one notch. Keep repeating with different sized clicks and notches until you step yourself along the entire curve. You make the clicks and notches small enough for "smooth" results, but big enough that you don't need a zillion of them.

Better yet, do your clicking and notching at the same time, by using a *vector* or *stroke* method that's able to approximate your curve with straight line segments. This looks a lot better than the step method, but there are still joint breaks. And an incredible amount of detail work is needed.

A fancier approach is to use little broken pieces of parabolas. This is the *quadratic spline* or *second order curvefit* method. But lots of pieces are still needed, and the results flat out do not look all that great. Which is one of many reasons that TrueType fonts are inferior to PostScript fonts. More on this another time.

I feel the "best" method for simple and graceful curves involves using a *cubic spline* technique.

There are two ways of looking at cubic splines, and you'll have to use both of them for full control. You can work in your *graph space*, where you are actually looking at the spline. Or you can work in your *equation space*, where you can precisely control your underlying math.

Figure one first shows you a cubic spline in its graph space. Regardless of its size, only *four* points (or eight x-y data values) are needed to fully specify the spline…

The *initial point* at x_0, y_0 tells you where the spline *starts*.

The *final point* at x_3, y_3 tells you where the spline *ends*.

The *first influence point* at x_1, y_1 should set the *direction* and *enthusiasm* with which your curve *leaves* the initial point.

The *second influence point* at x_2, y_2 should set the *direction* and the *enthusiasm* that your curve *enters* the final point.

Other names for the enthusiasm are the *tension* or the *velocity*.

A line connecting the initial point and first influence point defines the initial *tangent* direction used by the spline. The first tiny step along your spline should *always* head out in this direction. Similarly, a line connecting the second influence point and the final point defines the final *tangent* direction for your spline.

To build a cubic spline in the graph space, you set your end points. Then you move the influence points around to stretch and squeeze the spline into its intended shape. Typically, your influence points are well away from the curve itself.

Just like any middle management magician, you will find limits to the rabbits a single cubic spline can pull out of its hat. One spline can get you any straight line or smoothly curved path; some smooth curves having at most one *inflection* point (such as a sinewave); some curves with a single *cusp* (or "point"); or certain curves having one single and simple loop in them. All of which are both graceful and sparsely defined.

For fancier stuff, you use several splines end-to-end. We'll see more on this shortly.

Figure one also shows you a cubic spline in its equation space. Instead of relating *x* to *y*, you instead relate them *both* to a new *parameter t*. You could think of *t* as *time*. *t* will *always* range precisely from *zero* to *one* as your spline smoothly moves from its initial point to its final point.

One interesting way to look at the cubic splines is to think of a box in *x*, *y*, and *t* space. Inside the box is a three dimensional snake-like curve. Look in the *xy* end of the box and you will see your spline. Look in the *yt* side, and you will see how *y* varies as *t* (or time) goes from 0 to 1. Or look down through the *xt* top to see how *x* varies, again as *t* goes from 0 to 1.

Note that *x* and *y* are independent of each other. Each gets separately defined in terms of *t*. Knowing one does *not* reveal you the other. Unless you find *t* first. Do note that *t* usually does *not* move uniformly along your curve. Instead *t* tends to move faster along the "more bent" portions of the curve. And moves slower along your straighter portions.

Bopping between the equation and graph space is the key to controlling your splines. Figure one also shows you the simple ninth grade algebra needed to get from those influence points to the math equations, and vice versa. For some strange reason, this simple and obvious two-way math is very hard to find.

Let's repeat the basic cubic spline equations here…

$$x = \mathrm{A}t^3 + \mathrm{B}t^2 + \mathrm{C}t + \mathrm{D}$$
$$y = \mathrm{E}t^3 + \mathrm{F}t^2 + \mathrm{G}t + \mathrm{H}$$

Once again, *t* goes from zero to one as you go from the beginning to the end of your spline.

The obvious reason we call these cubic splines is that your highest power of *t* involved is a third power cube. The cubic term has most of its effect on the *final end* of your curve. The quartic (squared) term does most of its useful work in the *middle*. The linear term sets the initial *slope*, and the final D value sets your *offset*. By suitably combining these four terms, a wide and useful variety of spline curves can be generated.

But doesn't cubing sometime take forever on typical computers? Well, you can get sneaky and eliminate any need for raising anything to the third power. Instead of calculating each individual coefficient, what you do is rearrange the original equations as…

$$x = (((\mathrm{A}t) + \mathrm{B})t) + \mathrm{C})t + \mathrm{D}$$
$$y = (((\mathrm{E}t) + \mathrm{F})t) + \mathrm{G})t + \mathrm{H}$$

Or, restating things in English, to fake cubing, take A and multiply it by *t*. Add B. Multiply by *t* again. Add C. Multiply by *t* again. And, finally, add D. Presto. A cubeless cube.

Inside the machine or computer, the actual details of getting from any cubic spline to a frame bitmap or the tool paths will vary. But, in general, all you will do is repetitively use the simple algebra of figure one to decide where you go next. Consistent with your available step resolution.

Getting fancy

Although you might use any old computer language to explore cubic splines, I've found that the general purpose PostScript language is by far the fastest, easiest, and most fun way to play with Bézier curves. I've now posted a group of Bézier and cubic spline exploration tools to my *GEnie* PSRT. A few of these are also listed for you in figure one.

For instance, following a suitable scaling and translation, here is how you request the figure one spline in PostScript…

```
2 3 moveto
3 7.5 7 7 8 4 curveto
stroke
```

Note that only a few dozen bytes are needed to *fully* specify the spline, *regardless of its final size or device resolution!* This is sparse data used at its very best.

By the way, if you don't yet have real PostScript, we now do offer the shareware GHOSTSCRIPT emulator on PSRT. Which runs fake PostScript on nearly *any* printer or on nearly *any* computer screen. Yes, the full source code is included.

Does a cubic spline have a length? Well, obviously. But only if you are not a mathematician. Go through the length math, and a big ugly square root of some fourth order polynomial (which is *much* worse than it sounds) should leap out at you. After years of careful asking, I have yet to find *any* simple and exact closed form solution for the length of a cubic spline.

Yet, knowing the length of a spline gets *very* interesting for positioning typography on a curved surface, to fit fuzzy data, for correcting or creating distortions, or for that just-for-fun *avuncular sleezoid* surface you will find in figure two.

So, what I'll do instead is chop up any cubic spline into a hundred or so pieces. You then assume each piece is a straight line, and add their lengths up. Your answer is typically good enough for most graphical uses. And a mere hundred points usually gives you better than one part per thousand final accuracy.

By the way, if you do know how to quickly and conveniently find me the exact length of a cubic spline, please let me know. A free *Incredible Secret Money Machine II* for your trouble. Every math freak that I've talked to insists this is impossible. They have almost convinced me.

Things get interesting when you

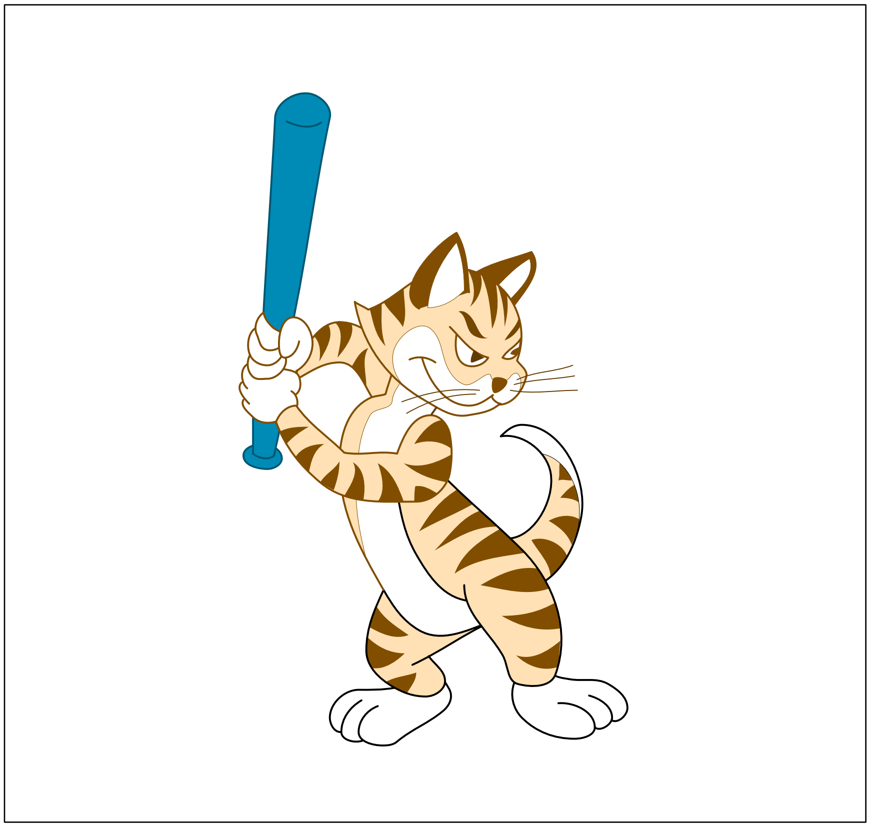


Fig. 3 – CURVETRACED CUBIC SPLINES are ideal for animation and cartoon characters, such as the "puss de resistance" shown here. Only a word processor is needed to create this quality artwork. No scanner is used.

SWITCHMODE RESOURCES

Allegro/Sprague
70 Pembroke Road
Concord, NH 03301
(603) 224-1961

Home Power
PO Box 130
Hornbrook, CA 96044
(916) 475-3179

Lambda Semiconductors
121 International Blvd
Corpus Christi, TX 78406
(512) 289-0403

Linear Technology
1630 McCarthy Blvd
Milpitas, CA 95035
(408) 432-1900

Maxim
120 San Gabriel Drive
Sunnyvale, CA 94086
(408) 737-7600

Motion
Box 6430
Orange, CA 92613
(714) 974-0200

Motion Control
800 Roosevelt Road E408
Glen Ellyn, IL 60137
(708) 469-3373

Motion Techniques
120 S chaparral Court #200
Anaheim, CA 92808
(714) 283-1123

Motorola
5005 E McDowell Road
Phoenix, AZ 85008
(800) 521-6274

National Semiconductor
475 Ellis Street
Mountain View, CA 94043
(800) 632-3531

PCIM
2472 Eastman Avenue #33-34
Ventura, CA 93003
(805) 658-0933

Power Techniques
120 S Chaparral Court #200
Anaheim, CA 92808
(714) 283-1123

SGS-Thomson
1000 East Bell Road
Phoenix, AZ 85022
(602) 867-6100

Siliconix
2201 Laurelwood Road
Santa Clara, CA 95054
(800) 554-5565

Texas Instruments
PO Box 401560
Dallas, TX 75240
(800) 336-5236

Unitrode Integrated Circuits
7 Continental Blvd
Merrimack, NH 03054
(603) 424-2410

splice cubic splines together. This is how you build fancy typography and other complex shapes. Ideally, you want to at least match your end points and your end point slopes. The rate of change of the curvature at each joint should also be constant, but this gets nasty in a hurry.

Picking the points gets tricky. Try to go too far with each spline and you lose accuracy. Don't go far enough, and you need too many splines and the splines may become very noise sensitive. Creating nervous results.

I've written a simple *curve tracing* routine that's now in my *PostScript Secrets* book/disk combo. You throw some endpoints and slopes at it, and it builds up all of your splines for you. Curvetracing is really great for high quality digitized signatures that need only tiny file sizes.

Curve tracing is also superb for animation. Figure three shows you *Meowwrrr*, our *Synergetics* puss de resistance. Curvetracing can give you precise and sharp results.

Only a word processor is needed; there's no scanning involved. I have included a full curvetracing toolkit in my #517 GONZO15A.PTL.

Ideally, you sure would like to just throw some fancy and noisy artwork at a cubic spline generator and have it automatically pick how many splines to use to produce the "best" possible results. This is not trivial, especially with noisy data.

You first have to decide how long each spline is to be and where it is to go. Then, you'll most often want to *constrain* all the intermediate splines, controlling both the entry angles and entry points. Or otherwise restricting them for continuity.

The "horses' mouth" paper on all of this is *Curvefitting with piecewise parametric cubics* by Michael Plass and Maureen Stone (with a little help by some guy named Warnock). This did appear in the July 1983 *Computer Graphics* pg. 229-239. And otherwise known as the SIGGRAPH Proceedings for 1983. The math here is unbearably heavy and obtuse.

So, I decided to take this general "fancy but noisy data into connected cubic splines" concept and have now generated somewhat similar results with #588 FUZZYBEZ.PS. Which will create the best smooth cubic spline through any noisy data while keeping your choice of constraints. Only ninth grade algebra is involved, and most of that gets fully automated. Speed is around five seconds per spline.

We will have several contests this month. To start off, either find me a simple Bézier length formula (lots of luck!), or show me a new and unusual task for cubic splines. Or curvetrace me something interesting.

There will be the usual *Incredible Secret Money Machine* book prizes, along with an all-expense-paid (FOB Thatcher, AZ) *tinaja quest* for two going to the very best of all.

Be sure to send your written entries to me at *Synergetics*, rather than to **Electronics Now** editorial.

Non-ionizing radiation safety

Do power lines cause cancer? One recent movie and a lot of my helpline callers do seem quite concerned over this. As with any controversy, your first step is picking up accurate and unbiased data. And useful tools for your own research.

Start off with *Polarized Debate: EMF's and Cancer*. Found in *Science* for December 11, 1992, p. 1724-1725. This is an unbiased summary. There's also the *Microwave News: A Report on Non-Ionizing Radiation*. This one is both expensive and has an alarmist tone about it.

As usual, any ongoing scientific controversy can be followed via the *Dialog Information Service*. And any newer important papers are likely to appear in an *Electric Power Research Institute* report.

For instruments and components, two sources include *F. W. Bell* and *Walker Scientific*. And ads for similar products appear in that *Compliance Engineering* trade journal.

One little noted fact: While high voltage power lines sound strong and nasty, their field strengths, expressed in *volts-per-mil* at sane distances, are *much lower* than your normal internal electrical human cell potentials. And overwhelmingly by far your largest magnetic field that you've ever been exposed to is the earth itself.

My own feelings are that if there were any major problems here, they would have become obvious a long time ago. There probably are some observable effects, but they probably do lie within acceptable risk bounds.

On the other hand, rethinking some really dumb stuff (such as building new playgrounds under high voltage lines or sitting on a police radar) is probably a good idea. And utilities studiously avoiding any careful and unbiased research certainly is not. There is probably a hacker buck to be made providing low cost monitoring instruments. And possibly a science fair project or school paper in your own surveys.

I do have a hunch this will prove to be an example of billions of dollars being blown on what will ultimately prove to be a monumentally large non-problem. There are much more urgent and far more important things to be doing with your time and effort.

National's simple switcher

The *National Semiconductor* folks have been offering free sample kits of their new *simple switcher* series of voltage regulators. Figure four shows you a typical circuit. Only five parts are needed to take a raw 7-60 volts of input and provide a fixed +5 volts of output. Other chips in the series give other values of fixed or adjustable output. In step-up, or step-down, or polarity inverting circuits.

We've looked at these switchmode regulators in previous columns and in the *Hardware Hacker* reprints. The major advantage of any switchmode regulator is in its potential efficiency. Stepping 40 volts on down to 5 volts with a one amp linear regulator is at its best only 12.5 percent efficient.

For each watt of load power, you have to burn up an extra seven watts internally.

But a switching regulator can, in theory, be 100 percent efficient. And getting above 85 percent is very easy in the real world. Thus, you only burn up around 150 milliwatts internally for each watt of load power. Which means less input power and less heat to get dumped. Heatsinks can often be eliminated entirely.

Any switching regulator does just that. It is a high speed switch that is rapidly turned on and off at a chosen duty cycle. Your duty cycle sets the

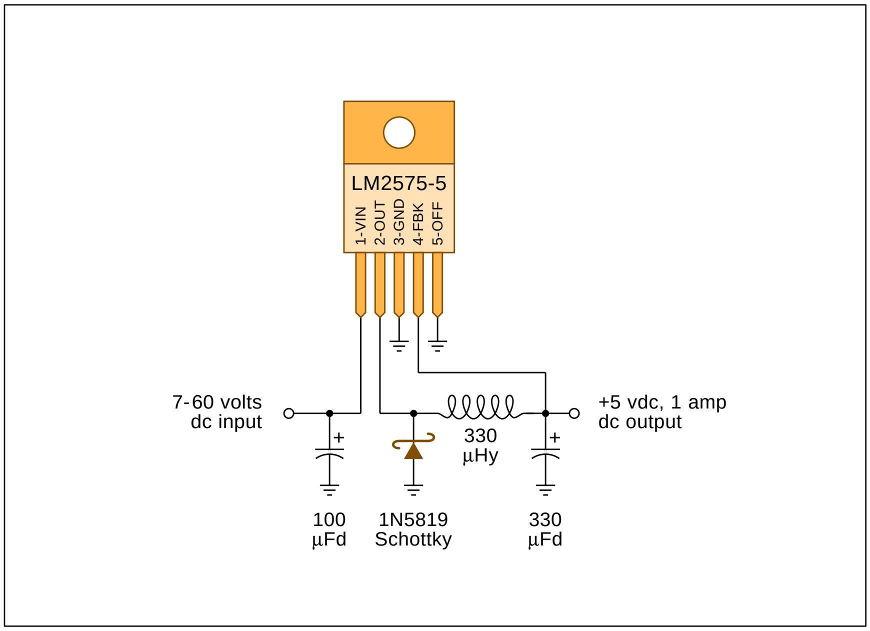


Fig. 4 – THE SIMPLE SWITCHER series of chips from National let you build an efficient switchmode regulator with only five parts. Heatsinking can often be eliminated completely. Free kits are available on professional request.

NAMES AND NUMBERS

F W Bell
6120 Hanging Moss Road
Orlando, FL 32807
(407) 678-6900

Compliance Engineering
593 Massachusetts Avenue
Boxborough, MA 01719
(508) 264-4208

Data I/O
P.O. Box 97046
Redmond, WA 98052
(800) 3-DATAIO

Dialog
3460 Hillview Avenue
Palo Alto, CA 94304
(415) 858-2700

EDN Asia
275 Washington Street
Newton, MA 02158
(617) 964-3030

Electric Power Research Inst
PO Box 10412
Palo Alto, CA 94303
(415) 855-2000

GEnie
401 N Washington Street
Rockville, MD 20850
(800) 638-9636

Gyration
12930 Saratoga Avenue, Bldg C
Saratoga, CA 95070
(408) 255-3016

Hong Kong Trade Council
219 East 46th Street
New York, NY 10017
(212) 838-8688

Microwave News
POB 1799, Grand Central Station
New York, NY 10163
(212) 517-2800

National Semiconductor
475 Ellis Street
Mountain View, CA 94043
(800) 632-3531

Philips
811 East Arques Avenue
Sunnyvale, CA 94088
(800) 447-1500

Science
1333 H Street NW
Washington, DC 20005
(202) 326-6400

Sound & Communications
25 Willowdale Avenue
Port Washington, NY 11050
(516) 767-2500

Sound & Video Contractor
9800 Metcalf
Overland Park, KS 66212
(913) 341-1300

Synergetics
Box 809
Thatcher, AZ 85552
(602) 428-4073

TDK
1600 Feehanville Drive
Mount Prospect, IL 60056
(708) 803-6100

Walker Scientific
Rockdale Street
Worcester, MA 01606
(800) 962-4638

average current through the output inductor, and thus your load current. A feedback loop adjusts how long the switch is on. During times the switch is off, a *freewheeling diode* continues the inductor's current path.

In our 40 volt input example, your switch is only on for one eighth of the time. Thus your average input current is only an eighth that of a comparable linear regulator. The frequency used is 50 kilohertz.

Full details appear on the related data sheets and ap notes. There's also a companion diskette that lets you optimize all your external component values. By the way, both the inductor and the freewheeling diode values are critical and *exact* brands and values *must* be used. The circuit simply will not work with an inductor having too high a dc resistance or one that ends up saturating at higher currents.

Similarly, ordinary silicon diodes are too slow and too inefficient.

Switchmode resources

For this month's resource sidebar, I have gathered together a few of the more obvious places to go for more information on these switching mode techniques. Not only for regulators, but also for motor controls and power inverters. The listing is mostly a mix of semiconductor houses who provide switchmode chips and a collection of the leading trade journals.

National's simple switcher is nice, but your best switchmode resource is *Maxim*, who have bunches of freebie samples and evaluation boards of an incredibly wide selection of voltage regulators and converters. Lots of ap notes, too.

Let me know if I've missed anyone important. There are scads of hacker opportunities here. Especially in the area of clean, stable, and lower cost power inverters. I'll try to have more circuits in future columns.

New tech lit

New *Electrorheological Fluids* are found in *Science* for October 30, 1992 on pages 761-766. These are fluids whose apparent viscosity can change with the applied voltage. Obvious aps would be clutches, 4WD-on-demand, and robotic muscles. Plus, of course, being a just-obscure-enough student paper topic.

From *Philips*, their *Semiconductor Sensors Data Handbook*. It includes magnetic field sensors, temperature sensors, and proximity detectors. And through *TDK*, a similar *Sensors* short form catalog on devices for infrared, humidity, current, surface potential, and even powder levels.

An enormous and free *Wall Chart of Programmable Devices* from *Data I/O* lists just about all the EPROM's, EEPROM's, PLD's, and such.

Two trade journals of interest to any professional sound installers are the *Sound & Communications* and the *Sound & Video Contractor*.

A reminder that I stock just about all of the important PostScript books from all of the major authors. I have even gathered together one each of everything into our PostScript *Whole Works* package. And much more on everything PostScript appears on my *GEnie* PSRT.

I've just added a new way to write fancy math equations in PostScript that includes some TEX-like features. See GNZOTRX2.PS for details. Or try HACK61.GPS for an editable copy of our figure one example.

Several helpline callers have asked for more info on *GEnie*. As always, you can call (800) 638-9636 for your local connect info. Because there are now thousands of nodes across the country, the chances are *very* good that *GEnie* access will be only a local call away. There are other low toll arrangements as well.

GEnie is not yet the largest of the information utility, but it is widely regarded as having the most and the best in the way of superb technical downloads. At last count, there were some 130,000 files on line waiting for your immediate use.

GEnie uses the "two tier" pricing structure. For a basic rate of $5 per month, you pick up unlimited email, movie reviews, the airline schedules, dial-a-joke, stock market quotes and great heaping bunches more. For the value-added services, such as PSRT and most of the computer libraries, there is a $6 per hour surcharge. On PSRT, this usually translates to an average of twenty one cents or so per library download.

Other *GEnie* RoundTables that you might just find of interest are RADIO, MAC, IBM, A2-PRO, DTP (desktop publishing) and HOSB (home office and small business). Most boards offer downloadable libraries, current news, and on-line conferences. Plus all of those usual BBS messages.

Two tips: If you are downloading a tutorial textfile, or a new PostScript routine that you are reasonably good at troubleshooting, use (L)ist rather than XMODEM. Assuming, of course, that you have a decent modem and clean lines. Listing an uncompressed textfile is faster and cheaper than XMODEMing a compressed version. Naturally, XMODEM (or its improved offspring) should still be used on any file that must be absolutely correct.

If you are PC based, check into the *Alladin* supervisor. This lets you call up in the middle of the night, check mail, read files, download libraries, and such. Much faster and far more conveniently than typing by hand.

Yes, *GEnie* is working on a few new graphical interfaces. But they still admittedly are ridiculously far behind on these.

UPDATE: I've left the *GEnie* stuff in this file as a historical record. They are no longer a major player. My PSRT files are slowly being revised and uploaded to my *Guru's Lair* website at *www.tinaja.com* as time and banner sponsors permit.

As usual, I've gathered much of the resources I've mentioned into the *Names and Numbers* and *Switchmode Resources* sidebars. Check here first before calling our tech helpline.

Speaking of which, you can get tech help, *consultant referrals, and off-the-wall networking by calling me at* (602) 428-4073. Uh, your best calling times are weekdays, Mountain Standard Time. ✥

NEED HELP?

Phone or write all your US Tech Musings questions to:

Don Lancaster
Synergetics
Box 809-EN
Thatcher, AZ, 85552
(520) 428-4073

US email: *don@tinaja.com*
Web page: *www.tinaja.com*