Conducted By David Newkirk, WJ1Z
Senior Assistant Technical Editor
e-mail: dnewkirk@arrl.org

Math in a Box, Transistor Modeling, and a New Meeting Place

> Someone told me that each equation I included in the book would halve the sales. I therefore resolved not to have any equations at all. In the end, however, I *did* put in one famous equation, $E = mc^2$. I hope that this will not scare off half of my potential readers.
>
> —Steven J. Hawking, in acknowledgments to his book *A Brief History of Time*

The TV you watch, or the radio transceiver which with you talk to the world, is a math machine—math put to work in the form of a kinetic sculpture built from copper, carbon, plastic, silicon and steel. You feed radio (or audio) and ac-line energy—both of which can be readily described numerically—in. The TV or radio performs mathematical operations on those inputs, and a desired, quantifiable result comes out: a sound, a moving picture, a declaration of rook to queen's bishop three from a chess opponent in Spain.

Because radio-electronic circuitry "just does math," math can predict and analyze the action of radio-electronic circuitry. Until recently, enjoying the fruits of heavier-duty radio math pretty much led to two choices: become fluent in enough math to do your own designing, or be content with someone else's designs. The first choice leads smack into the field of radio engineering, which isn't for everyone. The second choice amounts to little more than just consuming radio products engineered by someone else—another entirely acceptable approach to a radio hobby. Crossing from designer to user is relatively easy—just relax and float downstream, preferably without tools—but moving from user toward designer generally involves a growing struggle with math.

Math is actually a *language*—invariably a *second* language—that's greatly more abstract than any mother tongue. The snag is that, by tradition, the math education we receive in our formative years rarely reflects this fact. Our introduction to numbers, counting, addition, subtraction, multiplication and division proceeds smoothly (one might even say naturally) enough. But for many of us—probably most, if Steven Hawking's aside about equations in *A Brief History of Time* is a trustworthy indicator—a point eventually comes beyond which we just don't "get" how some string of alien, inorganic gibberish (say, $y = mx+b$) can possibly mean what its English supporting text says it means ("the definition of a line"). Remedial instruction fails, we may get another dose of that slow-acting poison—"*anyone* can do this if they *really* try" (struggling Morse code learners, does this ring a bell?)—real life or an acceptable substitute beckons, and we bolt for the door.

However sordid our educational pasts, many of us survive Amateur Radio license exams, slog our way through to a license, and find this hobby a blast. Some of us—by no means all of us—discover in ourselves that fiddling with what's *inside* a radio box can be quite a kick. Just building radios from scratch or kits may satisfy this discovery. To others, a little reading here and a bit of tentative calculator poking there—successfully scaling a 223-MHz antenna to 440 MHz, say—may suggest that the time for Peace Talks with Math may have come.

Suddenly, math's prickly language seems to be talking about things that *count*: deepening a TVI filter's stopband loss until the neighbors resume smiling, or flattening out an amplifier's phase response so it doesn't cream 9600-baud data. *Now* math's alien tongue refers to ideas we can wrap our minds around—problems that get past our math antibodies because *they're not just made up* like math-textbook examples. It's still tough, because it's just not our thing and probably never will be. But now we're going to dig in and get whatever value out of it we can.

This is a terrific time to make peace with math because this is the time of the personal computer. We use computers to do so many things that it's easy to forget that *all they ultimately do is math*. Whatever a computer does—e-mail, processing words, balancing a checkbook, stress-analyzing an aircraft wing, wafting you on a cybertrip down a virtual Amazon—it does by crunching numbers. If you can represent a thing or the action of a thing numerically—that is, if you can *model* it mathematically—you can simulate it and its action with a computer.

Since radio-electronic circuitry just does math, a computer can readily simulate, predict and analyze the action of radio-electronic circuitry.[1] What makes a computer a vastly more powerful tool for doing this than a slide rule or a simple calculator is that it can *contain* math in addition to just solving it. You can use a computer to determine, say, a coil's reactance (ac resistance) even if you know *absolutely nothing* about how to do it with math. All you do is run a program that contains the right equation(s) and lets you interact with them in a suitably friendly way. You tell the computer your coil's inductance and the frequency at which you need to know its reactance, and the computer tells you the answer.

This is why we're so excited about putting *ARRL Radio Designer*[2] into the hands of radio hobbyists. Quite a few hams would like to tinker with the nuts and bolts of radio-electronic circuitry in ways that their modest math fluency just doesn't allow. They know many of the issues, concepts and quantities involved in RF and electronic design, but aren't comfortable with manipulating them numerically. For them—as well as for math-fluent hams who want to spend less time doing busywork—*ARRL Radio Designer* is a natural.

I'm not saying that *ARRL Radio Designer* just hands over the keys to the radio-engineering castle for $150 plus shipping. What the program *does* do is stuff a large array of muscular radio-electronics math—math you can use regardless of how well you can speak it—into an affordable package. The more you already happen to know about radio and electronics design, the faster and farther you'll be able to go with *ARD*. Backing up your *ARRL Radio Designer* explorations with reading in solid textbooks like ARRL's reissue of Hayward's *Introduction to Radio Frequency Design* can brighten your light even faster.

More on Transistor Modeling in *ARRL Radio Designer*, Part 1

If modeling a circuit required us to *exactly* simulate the performance of the real thing, circuit simulators wouldn't be the major industrial and educational tools they are. Simulating even something as simple as a one-transistor 2-meter preamp would be so complex and painful that it wouldn't be worth doing. Luckily, much as the NTSC TV system can simulate moving pictures by tossing 525-line stills at us at the rate of 30 per second, simulated circuit behavior can be considerably grainier than a circuit's real-life performance and still return news we can use.

For example, in simulating a direct-conversion receiver's audio channel in March 1995 Exploring RF, we used *ARRL Radio*

[1] It gets better: Since radio-electronic circuitry just does math, and a computer just does math, a computer—properly configured and programmed, of course—can also do the *work* of radio-electronic circuitry. We call this *digital signal processing*, and it's an increasingly big deal.

[2] Available from ARRL Publications Sales for $150 plus shipping. See the ad elsewhere in this issue.

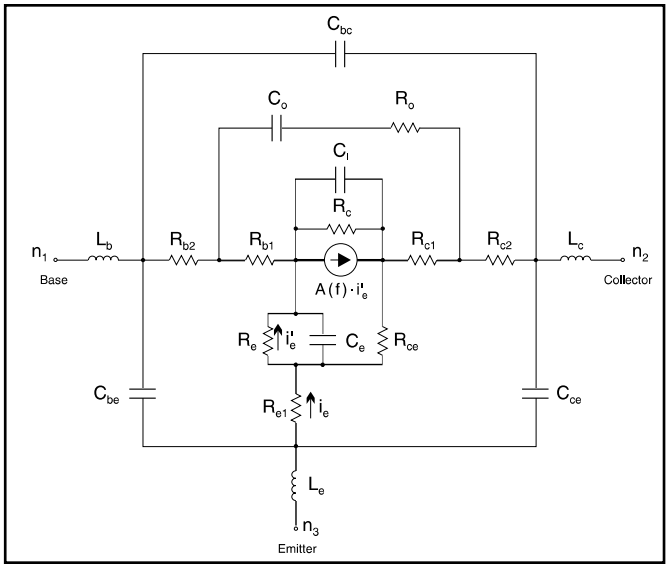

Figure 1—*ARRL Radio Designer*'s bipolar junction transistor (BIP) element allows you specify a BJT's internal structure in great detail. You can safely accept *ARD*'s defaults for many of these values when modeling at audio and low radio frequencies, but supplying true-to-life detail for accurate modeling at higher frequencies is a challenge in simulating with *ARD*. Most of the device characteristics listed in transistor databooks can't be directly plugged into this model!

Designer's BIP (bipolar junction transistor) element to supply only the roughest possible sketches of the circuit's four 2N3904 transistors.[3] But if you're already been thumbing through *The ARRL Radio Designer Manual*, you know that the BIP element can handle much more detail. Take a deep breath and look at Figure 1.

Precise active-device modeling lies at the heart of any circuit simulator worth using, so a radio engineer might wade right into the complexity of *ARD*'s BIP model with a bleat of pure joy. But many of us will wonder how we're ever going to model anything much more complex than a π attenuator if Figure 1 is any indication of what we'll need to succeed. A quick glance at the nearest transistor data manual seems to seal our doom: Of all the data listed for a particular type, almost none of it—if any—consists of numbers than we can plug directly into an *ARD* netlist!

Some circuit simulators—some of the *SPICE* variants come to mind—approach this problem by providing *device libraries* containing pre-coded device parameters. You select the part you want—by its actual part number, such as 2N5179, 2N3819 or 1N914—from a menu, and the program duly adds the corresponding code and device parameters to your netlist.

ARRL Radio Designer lets you do something a little different. Instead of device libraries, it comes with *device databanks*—manufacturer-supplied ASCII files that convey, in *S-parameter* (scattering parameter) form, the *actual measured performance* of RF devices operating under specific conditions. You can use this data directly or indirectly. The direct method involves plugging the databank data directly into *ARD*'s two-port [TWO] black-box element and using that TWO instead of a BIP or FET. The indirect method makes use of databank data and *ARD*'s optimization engine to adjust the values of a BIP or FET's many parameters until its performance matches the measured performance of an actual device as reflected in databank data.[4]

S-Parameter Basics

S parameters reflect a standardized way of characterizing how a device behaves in response to signal energy applied to its *ports*—its signal inputs and outputs—usually with all of its ports terminated in identical, standard impedances (commonly, 50 Ω resistive). A transistor, for instance, is a two-port device. By convention, the ports are labeled with numbers, Port 1 being the input and Port 2 being the output.

Signal energy applied to one port of a two-port device comes out two places: at the same port (the device reflects some of the energy back to the generator) and at the other port. How much signal comes out relative to the applied signal tells us the device's gain (which can be negative; that is, a loss); how much signal reflects back out tells us something about the impedance match between that port's impedance and our signal generator. Determining the phase of the output or reflection signals relative to the phase of the applied signals tells us even more about the device under test.

Figure 2 shows this idea graphically. An S parameter is a voltage ratio (commonly, but not always, expressed in decibels) annotated with two subscript numbers that indicate the ports involved. For instance, a device's forward transmission gain, S_{21}

[3] We—uh, that is, I—also supplied a first-rate howler when I wrote in that column that the *26* in the emitter resistance formula (R_E=26÷I_C) conveys "°C, that is—room temperature." Since that formula is just a variation on Ohm's Law, the *26* actually means 26 *millivolts*, the room-temperature value of V_T, which is the thermal equivalent of voltage in the transistor's semiconductor material.

[4] Although I've been mentioning only S parameters, *ARD* can also handle TWO data in Y or Z-parameter form.

Table 1

Two-Port S-Parameter Data Equivalent to an MRF586 Transistor (Operating at V_{CE}=10 and I_C=90 mA) from the *ARRL Radio Designer* Databank File MOTOROLA.FLP

```
100MHZ   .36   -123   15.68  107   .05   57   .44  -77
300MHZ   .33    180    5.78   83   .1    61   .32  -117
500MHZ   .34    154    3.44   70   .15   59   .39  -122
1000MHZ  .31    118    1.84   43   .25   51   .49  -133
```

Each dataset line conveys frequency and the S parameters MS_{11} (magnitude of S_{11}), PS_{11} (phase of S_{11}), MS_{21} (magnitude of S_{21}), PS_{21} (phase of S_{21}), MS_{12} (magnitude of S_{12}), PS_{12} (phase of S_{12}), MS_{22} (magnitude of S_{22}) and PS_{22} (phase of S_{22}).

Table 2

Two-Port S-Parameter Data Equivalent to a 2N3553 Transistor (Operating at V_{CE}=28 and I_C=30 mA) from the Motorola File 2N3553A.S2P

```
0.045   0.76   -165  10.39   96  0.03   38   0.23   -87
0.046   0.75   -166  10.09   95  0.03   46   0.23   -87
0.047   0.77   -166   9.84   94  0.03   48   0.22   -88
0.048   0.75   -166   9.78   94  0.02   44   0.22   -87
0.049   0.77   -166   9.67   94  0.03   45   0.22   -89
0.05    0.76   -167   9.39   94  0.03   51   0.21   -88
0.051   0.78   -168   9.23   93  0.03   51   0.21   -89
0.052   0.76   -168   9.04   93  0.03   49   0.21   -88
0.053   0.77   -167   8.92   92  0.03   52   0.2    -87
0.054   0.77   -168    8.7   92  0.03   44   0.21   -87
0.055   0.76   -168   8.66   92  0.03   49   0.21   -85
```

Like the .FLP-format data shown in Table 1, each of these lines conveys frequency and the S parameters MS_{11}, PS_{11}, MS_{21}, PS_{21}, MS_{12}, PS_{12}, MS_{22} and PS_{22}. The difference between this data format (that used by EESOF's *Touchstone* circuit simulator, and one of several S-parameter data formats now standard in the RF CAD community) and that of Table 1 is that an .S2P file conveys frequency in gigahertz but does not include frequency units. Because *ARD* assumes that you're talking hertz if you specify frequencies without units, you must add the suffix GHZ, with no leading space—`0.045GHZ` instead of `0.045 GHZ`—to each frequency spec in any .S2P data you use in an *ARD* netlist.

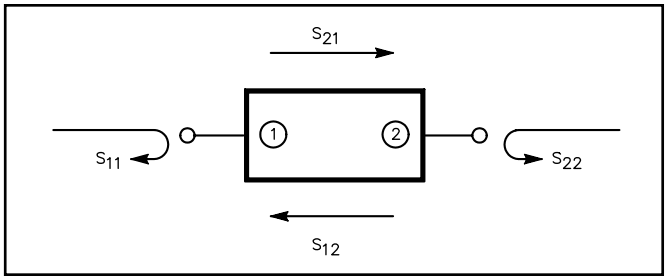

Figure 2—S parameters corresponding to input reflection, forward transmission gain, reverse transmission gain and output reflection can quite closely characterize a small-signal linear device—in this case, a *two-port* device. The text describes one way you can use *ARD*'s TWO element to insert manufacturer-supplied S-parameter data directly into your netlists.

("S sub two one"), is the ratio of the voltage at Port 2 to the voltage applied to Port 1—a value that must be expressed as a vector to convey the two signals' relative phase. To discuss S parameters more readily and to communicate them in tabular form, we split each of the four basic ones into separate magnitude and phase components: MS_{11} (magnitude of input reflection) and PS_{11} (phase of input reflection); MS_{21} (magnitude of forward transmission gain) and PS_{21} (phase of forward transmission gain); MS_{12} (magnitude of reverse transmission gain) and PS_{12} (phase of reverse transmission gain); and MS_{22} (magnitude of output reflection) and PS_{22} (phase of output reflection).

One big deal about S parameters is that they can cram a lot of very realistic information about a device's behavior into just a few standardized numerical values. If we know the S parameters of a given transistor operating under known conditions, we have a very useful picture of how it looks to the outside world—a picture we can paste directly into *ARRL Radio Designer*. This leads to another big deal about S parameters: RF-device manufacturers commonly make their products' S parameters freely available in forms that *ARD* can easily digest! Tables 1 and 2 show two very similar S-parameter formats that *ARD* can handle.

Plugging S-Parameters into ARRL Radio Designer

Using Table 1's data to synthesize a transistor in an *ARD* netlist is so straightforward that I won't even show you a complete netlist. Instead of connecting a BIP to nodes 3, 4 and 5, connect a TWO to those nodes in the same sense and give it a distinctive label:

```
TWO 3 4 5 MRF586
```

At the end of your file—after your FREQ block's END statement—add the TWO's S-parameter data in a DATA block:

```
DATA
 MRF586: S
 100MHZ  .36  -123  15.68  107  .05  57  .44 -77
 300MHZ  .33  180   5.78   83   .1   61  .32 -117
 500MHZ  .34  154   3.44   70   .15  59  .39 -122
 1000MHZ .31  118   1.84   43   .25  51  .49 -133
END
```

The data's header tags it as S-parameter information intended for use with an element labeled MRF586. Additional header information is necessary for Y or Z-parameter data, and for S-parameter data derived at impedances other than 50 Ω. (See pages 15-50 and 15-51 of *The ARRL Radio Designer Manual* for detail on the TWO element, and pages 17-7 and 17.8 of the *Manual* for how to specify data for TWO.)

This example is simplistic in that its DATA block doesn't include any numbers *ARD* can use to predict how a transistor-flavored TWO element may generate noise. We'll overcome this limitation in the next edition of Exploring RF.

ARRLCAD: A Meeting Place in Cyberspace

If you can send and receive Internet e-mail, you can now share your questions, answers, ideas and views about *ARRL Radio Designer* and other Amateur-Radio-related circuit and antenna design and simulation tools with other users. Thanks to the generous provision to ARRL of facilities and volunteer support by TAPR, Tucson Amateur Packet Radio Corp, I'm pleased to announce the startup of ARRL's computer-aided-design e-mail reflector, ARRLCAD.

You may already know of e-mail reflectors as *mailing lists*. A mailing list automatically sends copies of any messages sent in by its subscribers back out to all of its subscribers. It's somewhat like a UseNet newsgroup, the difference being that you don't need access to UseNet newsgroups to participate. All you need is the ability to send and receive Internet e-mail.

To subscribe to ARRLCAD, send an e-mail message to

```
listproc@tapr.org
```

with text that reads

```
subscribe arrlcad FirstName LastName
```

in which *FirstName* and *LastName* mean exactly that. (My subscription message read `subscribe arrlcad david newkirk`, for example.) The reflector software will confirm your subscription with a welcoming information message. Subscribing to ARRLCAD costs *nothing at all*.

***ARRL Radio Designer* Front-Line Support**

ARRL HQ's Technical Information Service provides front-line support for *ARRL Radio Designer*. Just ask for the Technical Information Service at 203-666-1541 (fax 203-665-7531), or send email to tis@arrl.org.

Free Electronic Goodies

This month's circuit example is so straightforward that it doesn't warrant the creation of an EXRF9505.TXT file, but as a consolation prize I've added to the heap the file ARD9410.TXT, an ASCII-text version of the October 1994 *QST* article that introduced *ARRL Radio Designer*. To find out how to get it and the files ARD.TXT, EXRF9501.TXT and EXRF9503.TXT from the ARRL Technical Information Service's automated info server (info@arrl.org), send the server an e-mail message that reads just

```
INDEX
HELP
```

and you'll receive instructions by return e-mail. ARD.TXT, ARD9410.TXT, EXRF9501.TXT and EXRF9503.TXT are also available from the ARRL BBS (203-666-0578) and via FTP from the ARRL subdirectory at oak.oakland.edu. QST

New Products

HAND-MADE KEYS/PADDLES

◊ If a hand-made key that won't slide around and won't be damaged by a fall to the floor is in your future, consider the line of keys and paddles produced by Karl Reiber, N6NXM, of Bakersfield, California.

Karl's keys, made one at a time by special order, come in various sizes and configurations and can be made from steel, brass and aluminum. They feature highly polished and tested needle bearings for smooth operation. Each piece is guaranteed for three years. If he can't repair your key, N6NXM will build you a new one.

Contact Titan Communications (ask for Jerry, KD6EAG) at 805-399-1321, or the designer, Karl Reiber, N6NXM, 446 Sycamore Dr, Bakersfield, CA 93308; 805-393-5855. QST