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

Transistor Modeling with *ARRL Radio Designer*, Part 2: Optimization Produces Realistic Transistor Simulations

As we saw in May 1995 *QST*'s Exploring RF, modeling a transistor's behavior merely by shoving its measured S, Y or Z parameters into *ARRL Radio Designer*'s two-port black-box element (TWO) is a neat thing to be able to do because it's so simple. All you do is link valid, transistor-flavored two-port data—a considerable body of which ships with each copy of *ARRL Radio Designer* in the form of manufacturer-supplied *databank* files, and more of which are ongoingly available from transistor manufacturers—into a TWO, and the black box acts like the transistor represented by the data.

Well, almost. Modeling transistors with the black-box-and-data approach involves a few limitations. For one thing, if you don't have S, Y or Z-parameter data for the transistor you want to model, you can't model it with a black-box element. Because manufacturers tend make data available only for relatively recent transistors intended for RF use, many garden-variety transistors don't show up in databank form. In other words, don't get your hopes up about finding manufacturer-supplied S-parameter data for, say, a 2N3904.

Even if you can find data for your device of interest, it may not be usable at your *frequency* of interest. For example, the *ARD* databank file MOTOROLA.FLP includes S-parameter data for an MRF581 operating at a collector current of 100 mA and with 10 V between its collector and emitter—

```
 300MHZ .66 -172 8.46 93 .05 49 .3 -134
 500MHZ .68 174 5.06 82 .07 56 .25 -147
1000MHZ .68 157 2.64 65 .12 64 .23 -172
1500MHZ .72 139 1.86 52 .17 63 .27 -177
```

—but only for 300, 500, 1000 and 1500 MHz. That none of these frequencies corresponds to a ham band is no problem: If a data set includes parameters for at least three frequencies, *ARRL Radio Designer* can interpolate parameter values for frequencies between them. All you need to do is specify the additional frequencies in your FREQ block.

A major limitation of databank data is that it cannot be reliably extrapolated— that is, used as a basis for educated guesses about device performance *outside* a data set's frequency span. It so happens, for instance, that many bipolar junction transistors intended for CATV service are just what the doctor ordered for strong, low-noise, linear amplifiers at HF. A broadband HF amplifier using a push-pull pair of MRF581s standing at 100 mA each would be a fine thing to model with *ARD*—but the lowest frequency for which Motorola supplies MRF581 S parameters is 300 MHz! What's a modeler to do?

Use *ARRL Radio Designer*'s BIP—bipolar junction transistor—model, that's what! As I wrote in the previous (May 1995) Exploring RF, however, a person just can't be expected to flesh out an *ARRL Radio Designer* BIP element—or an *ARD* FET for that matter, because it's just about as complex—by poring through transistor databooks and making educated guesses. The numbers we need to model transistors accurately just plain don't live in those books. Unless we're intimately involved with engineering and manufacturing transistors, we probably can't even begin to guesstimate appropriate starting values for a real transistor's tiny internal Rs, Cs and Ls—factors that are nonetheless crucial in modeling transistor behavior from the shortwaves on up.

The great news is that we don't have to guess at these parameter values. If we have valid S, Y or Z-parameter data for a given transistor, we can use *ARRL Radio Designer*'s optimizer to adjust a BIP's (or FET's) parameters so its behavior matches the data. The result is a realistic, databank-data-derived transistor model that can be used outside the frequency limits of the data used to generate it!

Table 1
***ARRL Radio Designer* Netlist for an Optimized BIP**

```
BLK
  BIP 1 2 0 A=0.98 RE=?1? F=?1GHZ? T=?0.01NS?
 + CE=?0.1PF? CI=?1PF? RC=?1000? RO=?0.1? CO=?1PF?
  + RB1=?1? RC1=?1E-6? RB2=?0.1? CBE=?10PF?
  + CBC=?0.1PF?
  + CCE=?100E-6PF?
  + LB=?1E-6NH? LC=?100E-6NH? LE=?0.1NH?
  OPTBIP:2POR 1 2 0
END
BLK
  TWO 1 2 0 MRF581
  REALBIP:2POR 1 2 0
END
FREQ
  300MHZ 500MHZ 1000MHZ 1500MHZ
END
OPT
  OPTBIP S=REALBIP
  TERM=0.01
END
DATA
  MRF581:S
 * MOTCL  MOTOROLA  MRF581 Vce=10V  Ic=100mA
   300MHZ .66 -172 8.46 93 .05 49 .3 -134
   500MHZ .68 174 5.06 82 .07 56 .25 -147
  1000MHZ .68 157 2.64 65 .12 64 .23 -172
  1500MHZ .72 139 1.86 52 .17 63 .27 -177
END
```

Data + Optimization = Realistic BIP

Doing this with *ARRL Radio Designer* is easier done than said. All we need is a netlist that includes:

- a BLK block containing the BIP element we want the optimizer to adjust
- a BLK block containing a TWO element that *ARD*'s optimizer can use as a standard of comparison in adjusting the BIP
- a FREQ block that specifies only the frequencies represented in our TWO's S (or Y or Z) parameter data
- an OPT block that tells *ARD*'s optimizer to compare the performance of the BIP's BLK to the performance of the TWO's BLK and tweak the BIP's parameters so their performances closely match; and
- a DATA block containing the S (or Y or Z) parameter data for the actual transistor we want our BIP to simulate.

Table 1 shows a ready-to-optimize netlist based on the MRF581 data presented a few paragraphs ago. Let's examine some of its finer points block by block.

The first block—

```
BLK
   BIP 1 2 0 A=0.98 RE=?1? F=?1GHZ? T=?0.01NS?
  + CE=?0.1PF? CI=?1PF? RC=?1000? RO=?0.1? CO=?1PF?
   + RB1=?1? RC1=?1E-6? RB2=?0.1? CBE=?10PF?
   + CBC=?0.1PF?
   + CCE=?100E-6PF?
   + LB=?1E-6NH? LC=?100E-6NH? LE=?0.1NH?
   OPTBIP:2POR 1 2 0
END
```

—contains the BIP we'll adjust for equivalence with databank data. All of its parameters are optimizable—question marks delimit optimizable values—except A (the BIP's alpha). (We must keep the optimizer's mitts off of alpha because it immediately attempts to raise alpha to a value greater than 1—impossible for a real transistor—and halts itself with a suitably pithy error message. [Try putting question marks around the A entry's 0.98 value when you optimize this netlist, and you'll see what I mean.])

Using the example of page 15-10 of *The ARRL Radio Designer Manual* as a guide, I've preloaded all of the BIP's optimizable parameters with nonzero values (zero-value parameters instantly choke the optimizer by violating the rules of floating-point math) at realistic orders of magnitude. Preloading at realistic orders of magnitude speeds optimization. Preloading values at wildly inappropriate orders of magnitude (10 farads for CE and 1 megohm for RB1, for instance) may simulate a transistor that's essentially broken from the gitgo, forcing the optimizer to dig itself out of such a deep hole that success may be impossible.

The + signs tell *ARD* to ignore hard carriage returns within the BIP specification. This lets us specify a BIP's many parameters without extending one long netlist line far off the edge of our screen.

We code our optimizable BIP into a circuit all by itself. For convenience, I've given this circuit the name OPTBIP because it contains the BIP we're going to optimize. (Any syntax-valid name is fine as long as we use the same one here and in the "circuit to optimize" part of our OPT block's first command line.)

The second block—

```
BLK
   TWO 1 2 0 MRF581
   REALBIP:2POR 1 2 0
END
```

—contains the TWO element to which *ARD*'s optimizer will compare the performance of the optimizable BIP in our OPTBIP circuit.

We code our referent TWO into a circuit all by itself. For convenience, I've given this circuit the name REALBIP because it's our means of injecting real bipolar-transistor performance, in the form of databank data, into this optimization. (There's nothing sacred about the name OPTBIP; any syntax-valid string is fine as long as we use the same name in the "goal" part of our OPT block's first command line.)

The MRF581 at the end of our TWO's netlist line is a *label* that lets us link databank data (contained in the DATA block and introduced by the same label) with this TWO. (The label could be any syntax-valid string; MRF581 is appropriate because we happen to be modeling one this time around.)

The third block—

```
FREQ
  300MHZ 500MHZ 1000MHZ 1500MHZ
END
```

—tells *ARRL Radio Designer* the frequencies at which we want to model (and optimize) the behavior of OPTBIP and REALBIP. This block's four frequencies mirror those of the MRF581 databank data.

The fourth block, the optimization block—

```
OPT
 OPTBIP S=REALBIP
 TERM=0.01
END
```

—tells *ARD*'s optimizer what to do. The first command line—

```
    OPTBIP S=REALBIP
```

—specifies our optimization goal, telling *ARD* to make the S parameters of the circuit called OPTBIP equivalent to those of the circuit called REALBIP. (I've kept this goal specification simple by not specifying *weighting*, through which we can tell *ARD* how much each of the optimizable circuit's S parameters [for a two-port network, S_{11}, S_{21}, S_{12} and S_{22}] contributes to the optimizer's error function. All four weights therefore default to 1.)

The next line—

```
 TERM=0.01
```

—tells *ARD* to terminate optimization when the optimizer's error function falls to 0.01. (If we see that the process has reached the point of diminishing returns before TERM has been reached, we can always abort the optimization and accept whatever results we have.)

This OPT block doesn't specify optimization frequencies, so the optimizer uses the frequencies specified in the FREQ block.

Finally, the DATA block—

```
DATA
  MRF581:S
* MOTCL   MOTOROLA  MRF581 Vce=10V   Ic =100mA
   300MHZ .66 -172 8.46 93 .05 49 .3 -134
   500MHZ .68 174 5.06 82 .07 56 .25 -147
  1000MHZ .68 157 2.64 65 .12 64 .23 -172
  1500MHZ .72 139 1.86 52 .17 63 .27 -177
END
```

—contains S-parameter data for a TWO element labeled MRF581—the TWO in the REALBIP block. This is the data to which the optimizer ultimately compares the optimizable BIP's performance during optimization.

Running It

Now comes the fun part. With the Table 1 netlist loaded into *ARRL Radio Designer*, click *ARD*'s **Optim** button or press **Shift+F10**. *ARD* analyzes the netlist and then pops its Optimization and Optimization Data dialog boxes. In the Optimization dialog, enter *100* as the number of iterations, change the optimization type to Gradient, clear the Display check box (a good habit to get into because turning off Display greatly speeds computations if you have any graphs onscreen), click Optimize (or press **Alt+O**), and you're off!

In just a few iterations, the optimizer pushes the error function below 0.2 and reports that the gradient has become too small for the process to continue. True, 0.2 is considerably higher than the TERM value we specified (0.01), but perhaps OPTBIP's performance already resembles that of REALBIP closely enough for whatever modeling we want to do. Take a look at Figures 1 and 2 and judge for yourself. It looks to me like OPTBIP's performance already falls comfortably within the sample-to-sample variation we'd expect in a handful of off-the-shelf MRF581s.

Getting Closer with Single-Frequency Modeling

We've just done a pretty good job of making an *ARD* BIP exhibit the gain and I/O-reflection behavior of a real transistor—over the frequency range (300 to 1500 MHz) covered by that transistor's databank data. Because the optimized BIP owes its performance to (among other things) the realistic internal resistance, capacitance and inductance values written into it by the optimizer, it's frequency-transportable in a way the REALBIP TWO's databank data isn't. We can now quite confidently use the optimized BIP at, say, 30 MHz—far below the frequency represented in the data set we used to optimize it!

If we want to use our optimized BIP *only* at frequencies below the data set's lowest frequency, we can improve our results by making the optimizer concentrate all of its number-crunching power at the lowest frequency in the data set. To do this, we just comment out the FREQ block's 500MHZ, 1000MHZ and 1500MHZ entries by inserting a semicolon between 300MHZ and 500MHZ, and add a leading asterisk to the corresponding lines in the DATA block's MRF581 data set.

Table 2 shows the results of doing this with our MRF581 data. The performance of OPTBIP and REALBIP now match much more closely at 300 MHz (error function ≈ 0.03) than we were able to achieve with the optimizer working to equalize their behaviors over the 300 to 1500-MHz range. Now we can get to work modeling that push-pull MRF581 amplifier at 14 MHz!

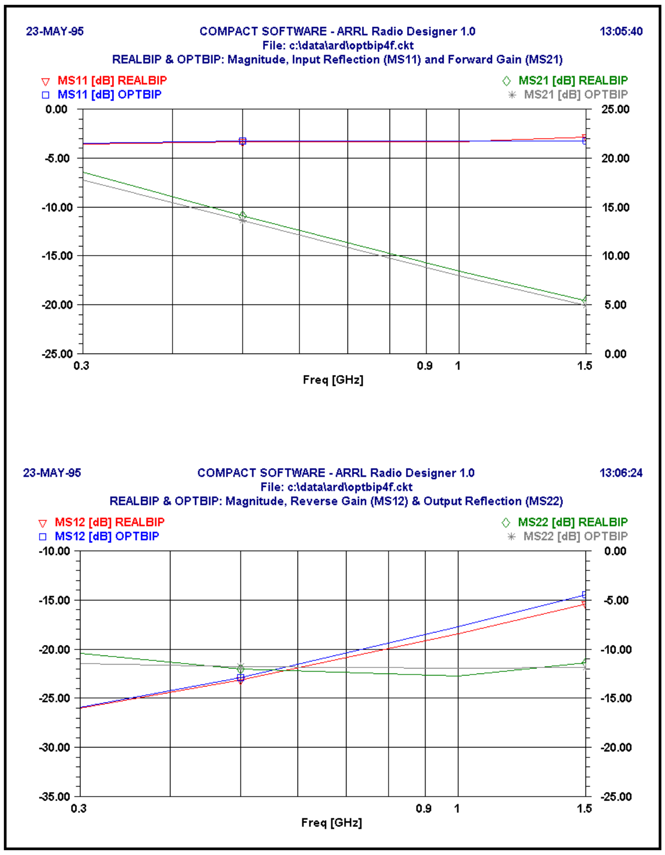

Figure 1—Optimized for a performance match with the measured S parameters of a real MRF581 transistor (`REALBIP` traces), the performance of an *ARRL Radio Designer* bipolar-junction transistor model (`OPTBIP` traces) mirrors the real thing to within a decibel. These graphs compare the magnitude components of the real and simulated transistors' S parameters; Figure 2 compares the phase components of both transistors' S parameters. (What's an S parameter, anyway? See Exploring RF for May 1995.)

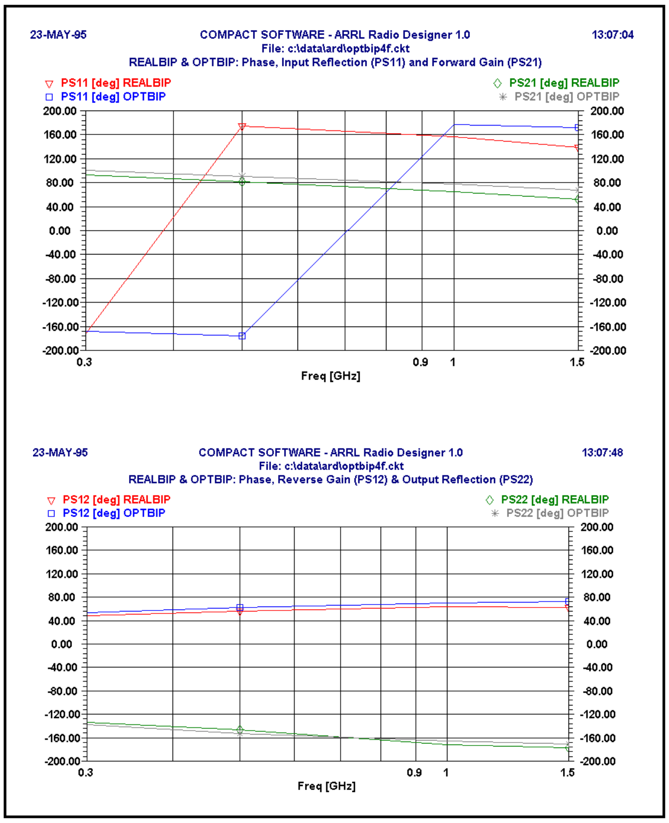

Figure 2—The phase performance of a real MRF581 transistor (`REALBIP` traces) and an *ARD* transistor model optimized to act like it (`OPTBIP` traces) coincides closely. The steep transitions in the PS_{11} and PS_{21} traces—from –180° to +180°—are artifacts of graphing these quantities rectangularly (–180° and +180° are actually the same point) with only four data points, so the PS_{11} and PS_{21} performance of `REALBIP` and `OPTBIP` track more closely than the wide horizontal gulf between the transition lines indicates to the eye.

What About Optimizing an *ARD* FET?

Now that I've shown you how to optimize a BIP on the basis of S parameters, you're ready to try a FET. *ARRL Radio Designer*'s Example 4, a simple JFET preamp, contains a FET element that's already optimized on the basis of the measured Y parameters of a 2N4416. Your job? Copy the Example 4 netlist into a new file and optimize an unoptimized FET element using the Y parameters in EXAMPLE4.CKT's `DATA` block.

Transistor Modeling with *ARD*, Part 3

Optimization is one of the two biggest smile-producing features *ARRL Radio Designer* brings to affordable RF CAD, and accurate noise modeling is the other. This month, I've shown you how you can crank a transistor's S, Y or Z parameters through *ARD*'s optimizer to make an *ARD* BIP or FET exhibit the same parameters, but space hasn't allowed me to touch on the complex issue of modeling noise with TWOs, BIPs and FETs. That important transistor-modeling concern deserves special treatment, and it's coming right up—in September *QST*'s Exploring RF.

Free Electronic Goodies

An expanded version of the Table 1 netlist is available (as the file EXRF9507.CKT) from ARRL HQ's Internet info server (send e-mail to **info@arrl.org** with a message text consisting of just the word *HELP*). A ZIP file containing EXRF9507.CKT and its accompanying *ARRL Radio Designer* report file (EXRF9507.RP2) is available from the HQ BBS (203-594-0306) as EXRF9507.ZIP, and via FTP (as exrf9507.zip) from the pub/hamradio/arrl directory at **oak.oakland.edu**.

ARRLCAD, an e-mail reflector (mailing list) through which 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, is up, running and growing. 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. (I subscribed with the message `subscribe arrlcad david newkirk`, for example.) The reflector software will confirm your subscription with a informational welcome message. Subscribing to and participating in ARRLCAD costs *nothing at all*. QST

Table 2
300-MHz S Parameters of `OPTBIP` and `REALBIP` (`OPTBIP` Optimized at 300 MHz)

	MS_{11} (dB)	PS_{11} (°)	MS_{21} (dB)	PS_{21} (°)	MS_{12} (dB)	PS_{12} (°)	MS_{22} (dB)	PS_{22} (°)
`REALBIP`	–3.61	–172.0	18.55	93.0	–26.02	49.0	–10.46	–134.0
`OPTBIP`	–3.23	–167.9	17.98	98.2	–26.31	53.6	–10.23	–132.9