SANDIA REPORT

SAND2018-14272
Unlimited Release
Printed December 2018

Application Note: Syntax, Parsing and Feature Differences Between HSPICE and *Xyce*™

Peter E. Sholander

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia, LLC.

SAND2018-14272
Unlimited Release
Printed December 2018

Application Note: Syntax, Parsing and Feature Differences Between HSPICE and *Xyce*™

Peter E. Sholander
Electrical Models and Simulation
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1177

Abstract

This application note describes some known differences in syntax, parsing, and supported features between the HSPICE and **Xyce** circuit simulators that might be relevant to both internal Sandia **Xyce** users and other performers on the DARPA Posh Open Source Hardware (POSH) program. It also presents strategies for converting HSPICE netlists and libraries to **Xyce** netlists and libraries.

Trademarks

Xyce Electronic Simulator™ and Xyce™ are trademarks of National Technology & Engineering Solutions of Sandia, LLC (NTESS).

All other trademarks are property of their respective owners.

Contact Information

Outside Sandia

World Wide Web	http://xyce.sandia.gov
Email	xyce@sandia.gov

Inside Sandia

World Wide Web	http://xyce.sandia.gov
Email	xyce-sandia@sandia.gov
Bug Reports	http://joseki-vm.sandia.gov/bugzilla http://morannon.sandia.gov/bugzilla

Contents

List of Figures

List of Tables

1. Introduction

Xyce is Sandia National Laboratories' SPICE-compatible high-performance analog circuit simulator, written to support the simulation needs of the laboratories' electrical designers. It has the capability to solve extremely large circuit problems on large-scale parallel computing platforms, and contains device models specifically tailored to meet Sandia's needs.

The Sandia electrial design community also uses many Commerical Off the Shelf (COTS) circuit simulators, such as HSPICE, PSpice, T-Spice and Spectre, as part of their design flow. Those designers often then want a "low-friction path" to convert netlists generated by those COTS tools into **Xyce** compatible netlists for subsequent use in Uncertainty Quantification (UQ) studies and/or analyses for unique environments.

The various "Spicen" that flowered from the original SPICE root-stock have all diverged with respect to their netlist languages, parser capabilities, and supported features. So, translation from one netlist format to another is not always a straightforward exercise. The translation process from PSpice to **Xyce** is discussed in the **Xyce** Reference Guide [1].

This application note documents differences between HSPICE and **Xyce** that might be relevant to both internal Sandia **Xyce** users and other performers on the DARPA Posh Open Source Hardware (POSH) program. It also presents strategies for converting HSPICE netlists and libraries to **Xyce** netlists and libraries. The differences noted herein were verified with **Xyce** Release 6.10 and HSPICE Version M-2017.03-SP2-1.

For external open-source users, source code for **Xyce** can be obtained from our website at xyce.sandia.gov. The **Xyce** Reference Guide [1] and Users' Guide [2] provide more detail on **Xyce** syntax and usage. They are also available at our website.

Finally, one purpose of this application note is to solicit feedback on additional incompatibilities and differences between **Xyce** and HSPICE. The **Xyce** development team can be contacted via email at xyce@sandia.gov. Feedback that includes small, runnable HSPICE netlists that illustrate the compatibility issues, and can be freely shared with other open source users, are especially helpful. Feedback allows the **Xyce** development team to prioritize future improvements to **Xyce**, based on requests from our internal users, our external partners, and our open-source user community.

2. Differences Between **Xyce** and HSPICE

This chapter discusses differences in syntax, parsing capabilities, and supported features between HSPICE and **Xyce**. The nominal focus is on the conversion of HSPICE netlists and libraries into **Xyce** compatible netlists and libraries.

2.1 Syntax and Parsing Differences

This section focuses on cases where **Xyce** and HSPICE are nominally “feature compatible”, but their netlist syntaxes and/or parsing capabilities are fundamentally different. So, the differences discussed in this section can mostly be resolved via character substitution, insertion and/or deletion.

2.1.1 Leading Whitespce

The netlist shown in Figure 2.1 runs correctly in HSPICE. However, **Xyce** parsing treats whitespace at the beginning of a netlist line as a comment character unless it is followed by +, which is the **Xyce** continuation symbol, in which case **Xyce** treats the line as a continuation. So, **Xyce** parsing of this netlist will exit with this error message (“There was 1 undefined symbol in .PRINT command: device V1”) because the V1 instance line was not processed by the **Xyce** parser. The work-around is to remove the leading whitespace.

```
Leading whitespace on a netlist line
************************************
 V1 1 0 DC=0 SIN(0 1 1e3 0 0)
R1 1 0 1
.TRAN 10us 1ms
.PRINT TRAN V(1) I(V1)
.END
```

Figure 2.1. HSPICE netlist with leading whitespace on a netlist line

2.1.2 Inline Comments

Hspice uses a dollar sign ($) to specify inline comments:

```
R1 1 0 1K $ Adding a 1K resistor to circuit
```

The equivalent way to specify an inline comment in Xyce is to use a semicolon:

```
R1 1 0 1K ; Adding a 1K resistor to circuit
```

2.1.3 Allowed Characters in Node and Device Names

Xyce and HSPICE have different rules for which characters are allowed in device names and node names. See Section 2.3.2 ("Legal Characters in Node and Device Names") of the **Xyce** Reference Guide [1] for more details on what is legal in **Xyce**.

2.1.4 Scaling Factors

The allowed scaling factors in **Xyce** are given in Chapter 4 of the **Xyce** Users' Guide [1]. There are at least two differences with HSPICE. First, the "atto" prefix, which is designated by "a" or "A", is acceptable in HSpice but is not accepted in **Xyce**. The use of the "atto" prefix in **Xyce** netlists must be replaced with "e-18".

In HSPICE, "x" and "X" are synonyms for "meg". So, all three scaling factors mean 1e6 in HSPICE. In **Xyce**, "x" and "X" are not recognized scaling factors. **Xyce** only uses "meg" to denote 1e6. So, this netlist line is legal in HSPICE but not in **Xyce**:

```
V1 1 0 SIN(0 1 1X 0 0)
```

2.1.5 Line Continuation Characters

HSPICE uses both "\" and "+" as line continuation characters, albeit in different ways. So, these (admittedly inartful) V1 and R1 instance lines are legal in HSPICE:

```
V1 1 0 \
SIN(0 1 1e3)
R1 1 2
+ 1
```

In Xyce, only “+” is used as a continuation character. So, the V1 instance line given above would fail Xyce parsing.

HSPICE also uses a double blackslash “\\” at the end of the line for continuation when the continuation is inside a token or string. An example of how to translate that into **Xyce** syntax will be given in Section 2.1.11.

2.1.6 Subcircuit Node Delineation and Wildcard Syntaxes

Hspice uses “.” to separate circuit-hierarchy levels whereas **Xyce** uses “:” for the same purpose. For instance, the Hspice syntax:

```
.PRINT DC V(X1.3)
```

which indicates that we wish to observe the voltage of node 3 in subcircuit X1 would have the equivalent **Xyce** syntax of:

```
.PRINT DC V(X1:3)
```

HSPICE allows the use of “wildcards” in `.PRINT` statements that reference subcircuit nodes. For example, this HSPICE syntax requests the nodal voltages at all nodes in subcircuit `X1`:

```
.PRINT DC V(X1.*)
```

That “subcircuit wildcard” syntax is not supported in **Xyce**.

2.1.7 Subcircuit Parameters

In **Xyce**, the preferred syntax uses the keyword `PARAMS:` to specify subcircuit parameters on both the `.SUBCKT` command line and on the X device instance line. An example is:

```
.SUBCKT RESISTOR 1 2 PARAMS: RESISTANCE = 1
R1 1 2 RESISTANCE
.ENDS
X1 1 0 RESISTOR PARAMS: RESISTANCE = 3
```

In Hspice, the `PARAMS:` keyword is not used, and the corresponding HSPICE netlist fragment would be:

```
.SUBCKT RESISTOR 1 2 RESISTANCE = 1
R1 1 2 RESISTANCE
.ENDS
X1 1 0 RESISTOR RESISTANCE = 3
```

For HSPICE compatibility, **Xyce** should accept either syntax. (Note: This issue was the subject of SRN Bug 1733.)

2.1.8 Expression Delimiters

Hspice delineates expressions via single quotes. An example HSPICE netlist fragment is:

```
.param r0 = 2
R1 1 0 '2*r0'
```

In the above netlist fragment, the resistance of `R1` is given by the expression `2*r0` which evaluates to 4. The corresponding preferred syntax in **Xyce** uses curly braces rather than single quotes.

```
.param r0 = 2
R1 1 0 {2*r0}
```

However, the HSPICE syntax (with the single quotes) should also work in all **Xyce** expression contexts. If not then replace the single quotes with curly braces in the **Xyce** netlist.

2.1.9 Curly Braces Around Simple Function Calls

Simple function calls in HSPICE do not require single quotes in order to be evaluated. For instance:

```
.param r = agauss(0,1,2)
```

defines a parameter `r` which calls the function `agauss` to retrieve a value. In **Xyce**, curly braces are recommend for the evaluation of every expression. The exceptions (per SRN Bug 1692) are cases where the **Xyce** expression:

- has no white space in it.
- has an equals sign, so that the expression is part of a form like this: `V=2.0*param`
- is not on a "command line" that begins with a dot (".").
- does not have a function (e.g., `agauss`) embedded in it.

The example HSPICE netlist line given above does not meet all of those criteria, and the **Xyce** syntax would need to be changed to:

```
.param r = { agauss(0,1,2) }
```

2.1.10 Nested Delimiters in Expressions

It is not clear that this is an issue when converting HSPICE netlists to **Xyce** netlists. However, it is not legal to have nested curly brackets in a Xyce expression. So, this set of netlist lines is not legal in **Xyce**:

```
.param a1 = 2
.param a2 = { 1 + {2*a1} }
```

The second line would have to be changed something like this:

```
.param a2 = { 1 + (2*a1) }
```

So, if an HSPICE expression contains nested single quotes then only the outermost ones should be changed into curly brackets in the corresponding **Xyce** expression. The inner ones should be changed into parentheses instead.

2.1.11 Continuation Characters in Expressions

As mentioned previously in Section 2.1.5, HSPICE uses a double blackslash "\\" at the end of the line for continuation when the continuation is inside a token or string. An HSPICE example within an expression is:

```
R4 4 0 R='r1\\
+r2'
```

The corresponding **Xyce** syntax would be this, where the first character of the second line is the **Xyce** continuation character "+" and the single quotes have been (optionally) replaced with curly braces to match the preferred **Xyce** syntax for expressions:

```
R4 4 0 R={r1
++r2}
```

2.1.12 User-Defined Functions

In HSPICE, `.PARAM` statements can be used to define functions. An example is this, which sets the resistance of device `R2` to 4:

```
.PARAM a1=2
.PARAM SQUARE(X) = 'X*X'
R2 2 0 'SQUARE(a1)'
```

The corresponding **Xyce** definition would use the `.FUNC` statement instead. So, that **Xyce** netlist fragment would be:

```
.PARAM a1=2
.FUNC SQUARE(X) {X*X}
R2 2 0 {SQUARE(a1)}
```

2.1.13 Special Variables

Xyce allows the use of `TIME` and `TEMP` to denote the current simulation time and temperature within expressions. HSPICE also uses `TIME`, but uses `TEMPER` instead of `TEMP`. The HSPICE special variable `HERTZ`, to denote the current simulation frequency, is not supported in **Xyce**. Finally, `VT` is a special variable in **Xyce** but not in HSPICE.

2.1.14 Ground Node Synonyms

HSPICE allows the use of either the number 0 or the names `GND`, `GND!` or `GROUND` as synonyms for the "ground node". **Xyce** can also treat 0, `GND`, `GND!` and `GROUND` as synonyms for the ground node. However, one `.PREPROCESS REPLACEGROUND` statement must be included in the **Xyce** netlist to enable that capability. Otherwise, **Xyce** will consider each of those four terms as separate nodes with only node 0 being treated as the ground node.

2.1.15 Global Nodes

The documentation for **Xyce** 6.10 states that global nodes have to start with the prefix "$G". That is actually not true. The `.GLOBAL` command was implemented in **Xyce** 5.1, for HSPICE compatibility, but was never documented in the **Xyce** Reference Guide or Users' Guide. So, the following **Xyce** netlist fragment defines a global node named G1, and then uses it in a subcircuit definition:

```
.GLOBAL g1
.subckt rsub  a  b g1
Rab  a  b  2
Rbg  G1  b  3
.ends
```

The documentation for the planned **Xyce** 6.11 release will include information on the `.GLOBAL` command.

2.1.16 Controlled Sources

Both HSPICE and **Xyce** support the E, F, G and H devices, which are the "controlled sources". There are many differences in the syntaxes and supported features for those devices in HSPICE and **Xyce**. So, this subsection focuses on examples that may be relevant to the DARPA POSH performers.

Both HSPICE and **Xyce** allow the E and G sources to have their outputs defined by an expression. However, their syntaxes can be different. HSPICE examples are as follows, where `VOL` is an allowed synonym for `VALUE` for the E source and `CUR` is an allowed synonym for `VALUE` for the G source in HSPICE:

```
E1 1 2 VOL='V(5)'
G1 3 4 CUR='V(5)'
```

The corresponding **Xyce** syntaxes are:

```
E1 1 2 VALUE=V(5)
G1 3 4 VALUE=V(5)
```

Some of these minor syntax differences, where the features are then identical, may be addressed in future **Xyce** releases.

2.2 OPTIONS Processing and Syntax

There are many feature differences between the "options" supported by HSPICE and **Xyce**. Some of the differences relevant to DARPA POSH are discussed in this section.

Feedback on additional HSPICE simulation options used by the DARPA POSH performers is welcome, since mapping between options in different circuit simulators is often not straightforward. (Note: the the **Xyce** Reference Guide [1] has a short discussion of this issue for PSpice and **Xyce**. Sometimes one PSpice option needs to be mapped into multiple **Xyce** options, or an option with the same name has a different meaning in those two simulators.)

2.2.1 .OPTION vs. .OPTIONS

HSPICE uses this format for specifying simulation options:

```
.OPTION <optionName> <value>
```

In **Xyce**, options for each supported package are called according to the following format, where `<pkg>` is a "package name" such as `DEVICE`. The "tag" is then similar to the HSPICE "optionName".

```
.OPTIONS <pkg> [<tag>=<value>]*
```

The use of an HSPICE-style .OPTION line in a **Xyce** netlist will typically produce a **Xyce** netlist parsing warning of the form "Unrecognized dot line will be ignored", rather than a parsing error. That **Xyce** warning message will contain the file name and line number. So, in general, it is important to check the **Xyce** warning messages when running netlists that have been translated from other Spicen.

2.2.2 .OPTION SCALE

In HSPICE, `.OPTION SCALE` scales geometric element instance parameters whose default unit is meters. This option is not currently supported by **Xyce**, but is viewed as a high-priority addition to a future **Xyce** release.

2.2.3 .OPTION BYPASS and .OPTION SIM_ACCURACY

Both HSPICE and **Xyce** have numerous options to control the tradeoffs between simulation accuracy and simulation run-time. This subsection discuss two HSPICE options that may be relevant to the DARPA POSH performers.

The HSPICE `.OPTION BYPASS` command "bypasses the model evaluations if the terminal voltages stay constant". It can be applied to MESFETs, JFETs, BJTs, MOSFETs and diodes. **Xyce** does not support bypass.

The HSPICE `.OPTION SIM_ACCURACY` allows the end-user to "set and modify the size of time steps. This option applies to all modes and tightens all tolerances, such as Newton-Raphson tolerance, local truncation error, and other errors". It has user-configurable values between 1 and 100, with a default value of 10. **Xyce** does not directly support this HSPICE option. However, the **Xyce** team has observed that similar tradeoffs can be obtained by adjusting its tolerances. It also appears that the default tolerances in **Xyce** will often produce more time-steps (and hence a longer run-time) than HSPICE when the default value of 10 is used for in HSPICE for `.OPTION SIM_ACCURACY`

2.3 Command Lines Not Supported in **Xyce**

The following command lines, found in HSPICE, are not directly supported in **Xyce**:

- .ALTER
- .TEMP
- .IF, .ELSEIF, .ELSE and .ENDIF

This list is not intended to be exhaustive, and the **Xyce** team solicits feedback on additional command lines that are useful for the DARPA POSH performers. For .TEMP, the corresponding **Xyce** approach is discussed in Section 2.4.15.

2.4 Features That Act Differently in **Xyce** vs. HSPICE

This section discuss features that act differently in HSPICE and **Xyce**. The focus is on ones that are of likely interest to the DARPA POSH program. So, this list is definitely not exhaustive.

2.4.1 Multiplicity (M Factor)

In HSPICE, the “multiplicity” (or “M Factor”) can be used to essentially specify multiple netlist devices in parallel via a single instance line. In **Xyce**, the terms “multiplicity factor” and “multiplier” are used to describe that same concept.

At present, the multiplicity factor (M parameter) is only supported in **Xyce** by the R, L, C and MOSFET device models and some BJT device models (VBIC 1.3 and MEXTRAM). It is not supported for the X device (subcircuits) which is a known deficiency for **Xyce** support of the DARPA POSH program.

2.4.2 .OP Output

The **Xyce** `.OP` output generally contains less information than the `.OP` output from SPICE3F5 and most other “Spicen”. The “.OP (Bias Point Analysis)” section of the **Xyce** Reference Guide [1] has a discussion of how to work around some of these limitations with the **Xyce** `.PRINT` output.

2.4.3 MOSFET Device Levels

The “levels” for various device models in **Xyce** may differ from those used in other circuit simulators. However, for HSPICE compatibility, **Xyce** will accept two model levels for these device models:

- BSIM3 is MOSFET levels 9 and 49
- BSIMSOI is MOSFET levels 10 and 57
- BSIM4 is MOSFET levels 14 and 54

2.4.4 Transient Source Functions

The **Xyce** Piecewise Linear (PWL) source is not fully compatible with the HSPICE implementation. The V and I device sections of the the **Xyce** Reference Guide [1] have a discussion of this issue.

The other source definitions (SIN, EXP and SFFM) are compatible but the HSPICE `perjitter` and `seed` parameters are not supported. **Xyce** also does not implement the HSPICE Pattern source or Single Frequency AM source.

2.4.5 Diode Model

There are several known differences between the HSPICE and **Xyce** diode models. Xyce has a Level 1 diode model which varies the saturation current I_S as a function of an optional area parameter. HSPICE has this model, along with an additional additive term that is a function of the sidewall area. In other words, the effective value of the saturation current in HSPICE, which we will denote here as I_s^{eff} is given by:

$$I_s^{\text{eff}} = I_s \times A_j + I_{sw} \times P_j$$

where I_s represents the standard saturation current per unit area, I_{sw} the sidewall saturation current per unit length, A_j is the junction area, and P_j is the junction perimeter. **Xyce** is missing the part of the model which incorporates I_{sw} and P_j (effectively treating both of them as 0).

Another issue is that HSPICE apparently allows multiple aliases (e.g., CJ, CJA and CJO for diodes) for a single model parameter. Not all of those aliases may be recognized by the **Xyce** parser. **Xyce** handles this by emitting a warning message during netlist parsing. For example, if an unrecognized parameter `BOGO` was used in a `.MODEL` statement for say a D1N3940 diode then **Xyce** parsing would emit the following warning message, "No model parameter BOGO found for model D1N3940 of type D, parameter ignored". However, the **Xyce** simulation would then run to completion.

One way to check for parameter name differences or unsupported parameters in any of the devices in your **Xyce** netlist, after conversion from HSPICE but before running a complete **Xyce** simulation, is to use the `-norun` command line option. That **Xyce** option does netlist parsing and syntax/topology analysis, and then exits before running the circuit simulation. The **Xyce** team can then help resolve any parameter mapping issues.

2.4.6 Device Types

The "device type" has several fundamental differences between HSPICE and **Xyce**. HSPICE uses the B-device to denote an IBIS (I/O Buffer Information Specification) buffer. The B-device is the non-linear dependent source in **Xyce**. The HSPICE S-device is an S parameter element. The S-device is a voltage controlled switch in **Xyce**. In HSPICE, the T-, U- and W-devices are all transmission line models. In Xyce, the U-device is a behavioral digital device, while the W-device is a current controlled switch. Finally, HSPICE has a P-device (port), which is not supported in **Xyce**.

2.4.7 Verilog-A Support

Xyce does have the capability to dynamically link in Verilog-A models. However, that capability is limited and not HSPICE compatible. In particular, it is not possible to insert Verilog-A models into **Xyce** via the netlist alone. So, **Xyce** does not support the HSPICE `.HDL` command.

2.4.8 Charge-based (or Charge-Conserved) Capacitor Model

The HSPICE capacitor device allows both C (capacitance) and Q (charge) as instance parameters. So, both of these instance lines are legal in HSPICE, where "expression" denotes a legal HSPICE expression:

```
C1 node1 node2 C='expression'
C2 node3 node4 Q='expression'
```

Xyce does not support Q as an instance parameter, and only the first instance line is legal in **Xyce**. The Q-based capability could likely be added though, if it is useful to the DARPA POSH performers.

2.4.9 AGAUSS and AUNIF

The `AGAUSS` and `GAUSS` functions are defined both in HSPICE and **Xyce** to handle Gaussian distributions. For uniform distributions, HSPICE then uses the `AUNIF` and `UNIF` functions, while **Xyce** uses the `RAND` function. The **Xyce** definitions are given in the "Expressions" section of the **Xyce** Reference Guide [1]. The HSPICE and **Xyce** versions of `AGAUSS`, `GAUSS`, and their respective `AUNIF`, `UNIF` and `RAND` functions, are *not* fully compatible yet. A summary of the issues is as follows.

- In HSPICE, if Monte Carlo (MC) sampling is *not* turned on then the `AGAUSS` and `GAUSS` functions just return the mean of the distribution. If MC sampling is turned on, then HSPICE will randomly sample the specified distributions.
- In **Xyce** 6.10, there is no mode that will result in the `AGAUSS`, `GAUSS` or `RAND` functions returning the mean. **Xyce** will always return a random number from the specified distribution.
- Currently, the `AGAUSS`, `GAUSS` and `RAND` functions in **Xyce** 6.10 are *not* connected to its sampling capability at all. Instead, the **Xyce** sampling code only samples parameters using its own separate specification and that code does not use the random functions in the **Xyce** expression library.

These `AGAUSS` and `AUNIF` compatibility issues are currently deemed a high priority for **Xyce**'s support of the DARPA POSH program.

2.4.10 Other Mathematical Functions and Operators

A number of mathematical functions exist in Hspice that either have different names in Xyce, or are simply not implemented in Xyce. Known issues of this nature are summarized in Table 2.1. The HSPICE functions that are not implemented in **Xyce** have empty entries in the **Xyce** column in that table. (Note: there are also capabilities in **Xyce** that are not in HSPICE , but this application note is focused on translating from HSPICE to **Xyce**.)

Table 2.1: Math and Control Functions in HSPICE and **Xyce**

HSPICE	Xyce	HSPICE Category	Description/Comments
&&	&	math	Boolean AND
||	|	math	Boolean OR
^	**	math	^ is an allowed synonym for the exponentiation operator (**) in HSPICE. However, ^ denotes boolean XOR in **Xyce**.

Table 2.1: Math and Control Functions in HSPICE and **Xyce**

HSPICE	Xyce	HSPICE Class	Description/Comments
nint(x)		math	Rounds x up or down, to the nearest integer.
def(x)		control	Returns 1 if parameter x is defined. 0 otherwise.
db(x)		math	returns the value of x in decibels
log10(x)	log(x) or log10x()	math	HSPICE returns the base 10 logarithm of the absolute value of x, with the sign of x: (sign of x)log10(abs(x))
log(x)	ln(x)	math	HSPICE returns the natural logarithm of the absolute value of x, with the sign of x: (sign of x)log(abs(x))
cond ?x : y	IF(cond,x,y)		Ternary operator returns x if cond is not zero. Otherwise, it returns y. An HSPICE example is .param z= 'condition ? x:y'
val(element)	element	various	returns a parameter value for a specified element. In HSPICE val(r1) returns the resistance value of the r1 resistor. In **Xyce**, just r1 is used.
val(element.param)	element:param	various	In HSPICE, val(r1.temp) returns the value of the temp parameter for resistor r1. In **Xyce**, just r1:temp is used.

2.4.11 Monte Carlo Analysis

HSPICE supports Monte Carlo sampling via the use of the `AGAUSS`, `GAUSS` and `RAND` keywords which are then used within expressions. The `.SAMPLING` command was added for the **Xyce** 6.10 release, but that capability is not compatible with how HSPICE does Monte Carlo sampling.

2.4.12 .TRAN syntax

The `.TRAN` command in **Xyce** only support single-point analyses. In addition, the command syntax differs between HSPICE and **Xyce**. In **Xyce**, the simplest form is (where the parameters enclosed in [ ] are optional):

```
.TRAN <initial step value> <final time value>
+ [<start time value> [<step ceiling value>]] [UIC]
```

In HSPICE, the simplest form is:

```
.TRAN <tstep1> <tstop1> [START=val] [UIC]
```

Both `START=val` and `<start time value>` refer to the time at which output of the simulation results begins, and both have a default of 0. In **Xyce**, the `<initial step value>` parameter is "used to calculate the initial time step". However, the `<tstep1>` parameter in HSPICE is the "printing or plotting increment for printer output and the suggested computing increment for post-processing". If the user desires **Xyce** output at fixed time intervals, then the appropriate `.OPTIONS OUTPUT` statement would produce interpolated output at the requested time points.

2.4.13 Using .AC Analyses with .TRAN

In some HSPICE simulations, a `.TRAN` analysis could be used to establish multiple operating points for a `.AC` analysis. This could be done, for example, with these HSPICE commands:

```
.TRAN 1n 5u $ Transient analysis
.OP 1u 2u 3u $ Request operating point analysis
.AC DEC 100 1 20e9 $ AC analysis
```

In this case, HSPICE would perform separate .AC analyses for all of the time values specified as well as one .AC run at time zero.

Xyce does not support this capability. Its `.AC` analysis is only done at time zero, and Xyce does not support the use of the `.TRAN` and `.AC` commands in the same netlist.

2.4.14 .NODESET

The **Xyce** `.NODESET` command uses a different strategy than either SPICE3F5 or HSPICE. So, the **Xyce** behavior may differ from that provided by `.NODESET` and `.OPTION DCHOLD` in HSPICE. In addition, **Xyce** does not allow the use of “wildcards” in `.NODESET` (or `.IC`) statements. The “.NODESET (Approximate Initial Condition, Bias point)” section of the **Xyce** Reference Guide [1] gives more details on the **Xyce** implementation.

2.4.15 .TEMP

The HSPICE `.TEMP` command allows the user to specify multiple temperatures, and the simulation will be repeated at each temperature. **Xyce** does not directly support `.TEMP`. Instead the desired simulation temperatures would be specified via the `.STEP` or `.DATA` commands in **Xyce** 6.10.

2.4.16 DTEMP

HSPICE supports the `DTEMP` instance parameter for its R, L and C devices. The HSPICE definition is “the temperature difference between the element and the circuit, in degrees Celsius, with a default value of 0. To modify the temperature for a particular element, use the DTEMP parameter in an instance line”.

Xyce does not support the `DTEMP` instance parameter for its R, L and C devices. Instead, those devices have a `TEMP` instance parameter which sets the device temperature. So, the work-around for **Xyce** would be to use its `TEMP` special variable (see Section 2.1.13). If, for example, an HSPICE R-device instance line had `DTEMP=<val>` then the corresponding **Xyce** instance line would use `TEMP={TEMP + <val>}`. The `DTEMP` instance parameter will likely be added for the R, L and C devices in a future **Xyce** release.

2.4.17 Power Calculations

The `P()` operator for power may give different results for semiconductor devices (D, J, M, Q and Z devices) and the lossless transmission device (T device) in **Xyce** than with HSPICE. The relevant device sections of the **Xyce** Reference Guide [1] give details on how power is calculated for each **Xyce** device.

2.4.18 .MEASURE

The **Xyce** implementation of `.MEASURE` has some incompatibilities and differences with HSPICE. See Section 2.1.14 of the Xyce Reference Guide [1] for more details. These differences will likely be resolved in a future **Xyce** release.

2.4.19 Noise Models and .NOISE Analyses

The `.NOISE` analysis capability in **Xyce** is not widely used yet. So, it is less “mature” than the `.TRAN` and `.DC` analysis capabilities in **Xyce**. One limitation is that the **Xyce** implementations of the noise models for some of the ADMS-derived semiconductor devices (such as VBIC) have not been validated yet. In particular, there was no “gold standard”, supplied by the Compact Model Coalition (CMC), for the VBIC device that the **Xyce** implementation could be tested against.

A second limitation is that the ability to print out the individual noise contributions, from all the noise sources in all of the devices in the netlist, is not an “officially supported” capability in **Xyce** 6.10. It is not offically supported because of the testing/validation issues mentioned above.

The third limitation is that not all **Xyce** devices support noise sources yet. Of particular interest may be the noise models for the BSIM4 device model. Also, the noise model for the FBH device model has not been implemented yet.

Another important difference is that **Xyce** and SPICE3F5 report root mean square (RMS) noise values. HSPICE reports mean square (MS) noise values.

2.4.20 Multiple .END Statements

The netlist shown in Figure 2.2 is legal in HSPICE. Both simulations will be run, once with the resistance of R2 equal to 2 and once with its resistance equal to 3. In **Xyce**, the simulation would only be run with the resistance of R2 equal to 2. All of the text after the first `.END` statement would be treated as comment lines by the **Xyce** parser. To run both simulations in **Xyce**, the appropriate `.STEP` or `.DATA` statement would be used to set the desired values for the resistance of R2.

```
Multiple .END statements
************************************
V1 1 0 SIN(0 1 1e3)
R1 1 2 1
R2 2 0 2
.TRAN 10u 1m
.PRINT TRAN V(1) V(2)
.END

V1 1 0 SIN(0 1 1e3)
R1 1 2 1
R2 2 0 3
.TRAN 10u 1m
.PRINT TRAN V(1) V(2)
.END
```

Figure 2.2. HSPICE netlist with multiple .END statements

3. HSPICE Compatibility Improvements in **Xyce** 6.10

This chapter discuss some improvements to HSPICE compatibility that are included in the **Xyce** 6.10 release.

3.1 Model Binning

An initial capability for “model binning” based on length (L) and width (W) parameters was added in **Xyce** 6.10 for MOSFET devices. The “.MODEL (Model Definition)” section of the **Xyce** Reference Guide [1] gives more details on how to invoke that new **Xyce** capability. It also describes some of the current limitations of that capability.

Additional known limitations are that the **Xyce** model binning is not currently compatible with parametric sweeps of L and W parameters. This issue was the subject of a question to the Google groups forum for **Xyce**:

`https://groups.google.com/forum/#!topic/xyce-users/rQ-R_N4NUYc`

Another limitation, which is also not mentioned in the **Xyce** Reference Guide, is that model binning is currently very generic, in that it only applies to the L and W parameters. However, some device models might use binning of other parameters, in addition to L and W.

3.2 .LIB Statements

Xyce does support the `.LIB` netlist line. However, in **Xyce** 6.8 and earlier (see SRN Bug 1731 and SON Bug 42), the parser would attempt to treat `.LIB <name>` statements the same as `.INC <name>` statements. The logic that supported that was flawed and was removed for **Xyce** 6.9 as SON Bug 980. So, **Xyce** 6.10 should now rigorously support the HSPICE `.LIB` syntax where there is a library definition statement (`.LIB <library name>`) and a library inclusion statement (`.LIB <library file> <library name>`). Incorrect use of `.LIB` should now return informative error messages.

As a final note, in correcting the `.LIB` behavior for **Xyce** 6.9, a small logic error was introduced that could cause issues with parsing library definitions when subcircuit definitions were inside them. That issue was corrected in the **Xyce** 6.10 release, as SON Bug 1038.

3.3 .DATA

Sweep loops in **Xyce** netlists can now be specified using an HSpice-style `.DATA` command. This capability, which has been applied to .STEP, .DC and .AC analysis in **Xyce** 6.10, allows the user to specify parameter loops in which multiple parameters are changed simultaneously. The ".DATA (Data Table for sweeps)" section of the **Xyce** Reference Guide [1] gives more details. At present, only the "inline" form of the `.DATA` command is implemented in **Xyce**. The "external file" and "column laminated" forms are not supported.

3.4 Device Model Optimization

The BSIM-CMG version 110 (MOSFET level 110) and the 3-terminal version of VBIC 1.3 (BJT level 11) have been extensively optimized and now run much faster than they did in previous **Xyce** releases.

References

[1] Eric R. Keiter, Karthik V. Aadithya, Ting Mei, Thomas V. Russo, Richard L. Schiek, Peter E. Sholander, Heidi K. Thornquist, and Jason C. Verley. Xyce Parallel Electronic Simulator: Reference Guide, Version 6.10. Technical Report SAND2018-12374, Sandia National Laboratories, Albuquerque, NM, 2018.

[2] Eric R. Keiter, Karthik V. Aadithya, Ting Mei, Thomas V. Russo, Richard L. Schiek, Peter E. Sholander, Heidi K. Thornquist, and Jason C. Verley. Xyce Parallel Electronic Simulator: Users' Guide, Version 6.10. Technical Report SAND2018-12373, Sandia National Laboratories, Albuquerque, NM, 2018.

DISTRIBUTION:

1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories