Application Note: Syntax, Parsing and Feature Differences Between HSPICE and Xyce™ 6.11

Peter E. Sholander
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. This version is focused on Release 6.11 of Xyce.
Trademarks

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

Contact Information

Address
Electrical Models & Simulation Dept.
Sandia National Laboratories
P.O. Box 5800, MS 1177
Albuquerque, NM 87185-1177

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

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

Bug Reports
http://joseki-vm.sandia.gov/bugzilla
http://morannon.sandia.gov/bugzilla
## CONTENTS

1. Introduction ........................................ 9

2. Differences Between Xyce and HSPICE ............. 10
   2.1. Syntax and Parsing Differences .................. 10
       2.1.1. Leading Whitespace .......................... 10
       2.1.2. Inline Comments ............................. 11
       2.1.3. Allowed Characters in Node and Device Names .. 11
       2.1.4. Scaling Factors ............................. 11
       2.1.5. Line Continuation Characters ................. 12
       2.1.6. Mismatched Parentheses in .MODEL Statements .. 12
       2.1.7. Subcircuit Node Delineation and Wildcard Syntaxes .. 12
       2.1.8. Subcircuit Parameters ....................... 13
       2.1.9. .MACRO as a Synonym for .SUBCKT ............ 13
       2.1.10. Expression Delimiters ....................... 14
       2.1.11. Curly Braces Around Simple Function Calls .... 14
       2.1.12. Expression Delimiters Embedded in Function Calls .. 15
       2.1.13. Continuation Characters in Expressions ...... 15
       2.1.14. User-Defined Functions ..................... 15
       2.1.15. Special Variables ........................... 16
       2.1.16. Ground Node Synonyms ....................... 16
       2.1.17. Controlled Sources ......................... 17
       2.1.18. Independent Sources ......................... 17
   2.2. OPTIONS Processing and Syntax .................. 17
       2.2.1. .OPTION vs. .OPTIONS ...................... 18
       2.2.2. .OPTION SCALE ............................. 18
       2.2.3. .OPTION BYPASS and .OPTION SIM_ACCURACY .... 18
   2.3. Command Lines Not Supported in Xyce ............ 19
   2.4. Features That Act Differently in Xyce vs. HSPICE .. 19
       2.4.1. Multiplicity (M Factor) ....................... 19
       2.4.2. .OP Output ................................... 19
       2.4.3. MOSFET Device Levels ......................... 20
       2.4.4. Model Binning ................................ 20
       2.4.5. Transient Source Functions .................. 20
       2.4.6. Solution-Dependent Resistors ................. 20
       2.4.7. Diode Model ................................ 21
       2.4.8. Device Types ............................... 21
       2.4.9. Verilog-A Support ......................... 22
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>HSPICE netlist with leading whitespace on a netlist line</td>
<td>10</td>
</tr>
<tr>
<td>2-2</td>
<td>HSPICE netlist with multiple .END statements</td>
<td>27</td>
</tr>
<tr>
<td>3-1</td>
<td>Example Netlist for Xyce .TRAN Analysis</td>
<td>29</td>
</tr>
<tr>
<td>3-2</td>
<td>Example Netlist for Xyce .DC Analysis</td>
<td>29</td>
</tr>
<tr>
<td>3-3</td>
<td>Example Netlist for Xyce .AC Analysis</td>
<td>30</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 2-1. Math and Control Functions in HSPICE and Xyce ......................... 23
Table 5-1. -hspice-ext Command Line Option ........................................... 37
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 electrical design community also uses many Commercial 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 Release 6.11 of 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 6.11 and HSPICE Version M-2017.03-SP2-1. A previous version of this application noted covered compatibility between HSPICE and Xyce Release 6.10 [2].

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 [3] 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 Whitespace

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.

```
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

The following three syntaxes have also been observed to be either comment lines or inline comments in HSPICE. They should be replaced with the appropriate Xyce comment or in-line comment syntax.

R1 1 0 1K // double slash functions as an in-line comment
// double slash can also start a comment line
R1 1 0 1K * star outside of an expression functions as an in-line comment

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]. The main difference is that the “atto” prefix, which is designated by “a” or “A”, is acceptable in HSpice but is not accepted by default in Xyce.

There are now two options for handling this. The “atto” prefix in HSPICE netlists, that will be run in Xyce, can be manually replaced with “e-18” with a text editor. Alternately, the new -hspice-ext command line allows the user to toggle between the normal Xyce usage of “A” not being a scaling factor and having “A” be the “atto” scaling factor. See Section 5.2 for more details.
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 2.1.13.

2.1.6. Mismatched Parentheses in .MODEL Statements

The use of parentheses around the model parameters in a .MODEL statement is optional in both HSPICE and Xyce. However, if parentheses are used in Xyce then a matching pair must be used. The author has seen this syntax used in HSPICE netlists, where the trailing parenthesis is present but lacks the line continuation character before it.

```
.MODEL RMOD R (RSH=1
```

This HSPICE syntax will fail Xyce netlist parsing, and may cause unpredictable error/warning messages from the Xyce parser.

2.1.7. 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.

### 2.1.8. 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.)

### 2.1.9. .MACRO as a Synonym for .SUBCKT

In HSPICE, `.MACRO` and `.EOM` are synonyms for `.SUBCKT` and `.ENDS`, respectively. Xyce does not support the `.MACRO` and `.EOM` synonyms.
2.1.10. 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.

2.1.11. 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)
.param t = limit(-1,1)
```

define the parameters r and t which call the functions agauss and limit to retrieve values. 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 lines given above do not meet all of those criteria, and the Xyce syntax would need to be changed to:

```
.param r = { agauss(0,1,2) }
.param t = { limit(-1,1) }
```
2.1.12. Expression Delimiters Embedded in Function Calls

In HSPICE, an expression delimiter can be embedded with a function call, as follows where the parameter p2 is defined elsewhere in the netlist:

\[ .\text{param} \ p1 = \text{agauss}(\text{’1-p2’},1,3) \]

Per the previous subsection, the \text{agauss} call must be enclosed within expression delimiters in a Xyce netlist. However, the expression delimiters within the \text{agauss} function call must then be removed. So, the corresponding Xyce syntax would be:

\[ .\text{param} \ p1 = \{ \text{agauss}(1-p2,1,3) \} \]

2.1.13. 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:

\[ R4 \ 4 \ 0 \ R={r1+++r2} \]

2.1.14. User-Defined Functions

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

\[ .\text{PARAM} \ a1=2 \]
\[ .\text{PARAM} \ 	ext{SQUARE}(X) = ’X*X’ \]
\[ R2 \ 2 \ 0 \ ’\text{SQUARE}(a1)’ \]

The corresponding Xyce definition would use the .\text{FUNC} statement instead. So, that Xyce netlist fragment would be:
This particular issue can become complicated since this netlist line is legal in HSPICE, where the .PARAM line contains a mix of parameter definitions and function definitions:

```
.PARAM a1=2 a2=3 SQUARE(X) = 'X*X' a3=4
```

It could be translated into Xyce as follows:

```
.PARAM a1=2 a2=3 a3=4
.FUNC SQUARE(X) {X*X}
```

Similarly, multiple functions definitions on one HSPICE .PARAM line would have to be broken out into separate .FUNC lines in the translated Xyce netlist.

This particular translation issue will usually produce a Xyce error message similar to “Equal sign (=) required between parameter and value in .PARAM statement”, with a reference then to a specific file name and line number. So, this translation issue may be complicated to fix via scripting but is typically easy to diagnose.

### 2.1.15. Special Variables

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

### 2.1.16. Ground Node Synonyms

HSPICE allows the use of either the number 0 or the names \texttt{GND}, \texttt{GND!} or \texttt{GROUND} as synonyms for the “ground node”. Xyce can also treat 0, \texttt{GND}, \texttt{GND!} and \texttt{GROUND} as synonyms for the ground node. However, one .PREPROCESS \texttt{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.
2.1.17. 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. Of particular relevant to the DARPA POSH performers was that $V_0L$ is an allowed synonym for $VALUE$ for the E source and $CUR$ is an allowed synonym for $VALUE$ for the G source in HSPICE. Those keywords were not supported in Xyce 6.10, but are now supported in Xyce 6.11 for the E and G sources.

The following syntaxes are legal for E and G sources in HSPICE, where the $VCVS$ and $VCCS$ keywords are optional. Those keywords are illegal for Xyce E and G sources and must be elided from the corresponding Xyce instance lines.

E0 1 2 VCVS 3 4 1
G0 4 5 VCCS 6 7 1

2.1.18. Independent Sources

Both HSPICE and Xyce support the I and V devices, which are the “independent sources”. There are differences in the syntaxes and supported features for those devices in HSPICE and Xyce. Of particular relevance to the DARPA POSH performers is a trivial difference for DC sources. HSPICE uses an equal sign between the optional DC keyword and its value, while Xyce does not. The HSPICE syntax is shown below.

V1 2 0 DC=5

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 2.4.18.

2.4. FEATURES THAT ACT DIFFERENTLY IN XYCE VS. HSPICE

This section discusses 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. **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 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. Another limitation, which is not explicitly 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.

2.4.5. **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 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.

2.4.6. **Solution-Dependent Resistors**

In HSPICE, it is legal to have the resistance value for an R device depend on a solution variable. An example is as follows, where P1 is a parameter defined elsewhere in the netlist:

```
R1 1 2 'P1*v(in,out)'
```
Xyce does not support solution-dependent resistors. So, the R device instance line shown above would have to be converted to a Xyce Nonlinear Dependent Source (B device). The Xyce Users’ Guide [3] provides more details on “Analog Behavioral Modeling”. However, an example translation is as follows. This translation does not add any additional branch currents to the netlist, but it would not work if the R value could be zero.

```
BR1 1 2 I=’V(1,2)/(P1*v(in,out))’
```

### 2.4.7. 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 $I_{S_{\text{eff}}}$ 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.

### 2.4.8. 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.
2.4.9. 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.

2.4.10. 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 was no mode that would result in the AGAUSS, or GAUSS functions returning the mean. Xyce would always return a random number from the specified Gaussian distribution. The new -hspice-ext command line option now provides that “toggling” capability. See Section 5.2 for more details.

- Currently, the AGAUSS, GAUSS and RAND functions in Xyce 6.11 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.

2.4.11. 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.)

For user convenience, the new -hspice-ext command line option allows the user to toggle between the Xyce and HSPICE definitions for various commonly used math operators. See Section 5.2 for more details.
<table>
<thead>
<tr>
<th>HSPICE</th>
<th>Xyce</th>
<th>HSPICE Category</th>
<th>Description/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>&amp;</td>
<td>math</td>
<td>Boolean AND</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>~</td>
<td>**</td>
<td>math</td>
<td>^ is an allowed synonym for the exponentiation operator (** in HSPICE. However, ^ denotes boolean XOR in Xyce.</td>
</tr>
<tr>
<td>nint(x)</td>
<td></td>
<td>math</td>
<td>Rounds x up or down, to the nearest integer.</td>
</tr>
<tr>
<td>def(x)</td>
<td></td>
<td>control</td>
<td>Returns 1 if parameter x is defined. 0 otherwise.</td>
</tr>
<tr>
<td>db(x)</td>
<td></td>
<td>math</td>
<td>returns the value of x in decibels</td>
</tr>
<tr>
<td>log10(x)</td>
<td>log(x) or log10x()</td>
<td>math</td>
<td>HSPICE returns the base 10 logarithm of the absolute value of x, with the sign of x: (sign of x)log10(abs(x))</td>
</tr>
<tr>
<td>log(x)</td>
<td>ln(x)</td>
<td>math</td>
<td>HSPICE returns the natural logarithm of the absolute value of x, with the sign of x: (sign of x)log(abs(x))</td>
</tr>
<tr>
<td>val(element)</td>
<td>element</td>
<td>various</td>
<td>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.</td>
</tr>
<tr>
<td>val(element.param)</td>
<td>element:param</td>
<td>various</td>
<td>In HSPICE, val(r1.temp) returns the value of the temp parameter for resistor r1. In Xyce, just r1:temp is used.</td>
</tr>
</tbody>
</table>
### 2.4.12. 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 6.11 release, but that capability is not compatible with how HSPICE does Monte Carlo sampling.

### 2.4.13. .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.

### 2.4.14. 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.
2.4.15. .DATA

As of Release 6.10, sweep loops for .STEP, .DC and .AC analyses can be specified in Xyce using an HSPICE-style .DATA command. 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. The “external file” and “column laminated” forms are not supported in Xyce.

2.4.16. .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.

2.4.17. Nominal Temperature

The default nominal temperature (TNOM) in HSPICE is 25°C. The default nominal temperature in Xyce is 27°C. The nominal temperature for all devices in a Xyce netlist can be set via this .OPTIONS line:

.OPTIONS DEVICE TNOM=<tempVal>

That global TNOM setting in Xyce can then be overwritten in various model cards, if needed.

2.4.18. .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 6.11.

2.4.19. 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 2.1.15). 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.

**2.4.20. 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.

**2.4.21. .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.

**2.4.22. 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” for noise 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 6.11. It is not officially 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.
2.4.23. 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. VIEWING XYCE OUTPUT FILES IN WAVEVIEW

This chapter provides an overview of how to view Xyce output files for .TRAN, .DC and .AC analyses, without the use of .STEP, in the WaveView tool. The guidance given below was verified with version 2016.06-SP1-1 of WaveView. Since there are valid ways to view the Xyce output in WaveView, the Xyce Team has not added a FORMAT=HSPICE parameter to any of the Xyce .PRINT lines yet. The summary is:

- The Xyce raw file output for .PRINT TRAN, .PRINT AC and .PRINT DC are all viewable in WaveView.

- Xyce-generated .prn files are viewable in Waveview for .PRINT TRAN and .PRINT DC, if FORMAT=NOINDEX is used. For .PRINT AC output, some manual editing of the header lines in the .prn files is required, as described below.

3.1. .TRAN OUTPUT

The .TRAN netlist shown in Figure 3-1 produces a testTran.cir.prn file that is viewable in WaveView. The only caveat is that WaveView may produce an error about “Signal V(1) has non-monotonic X values” before it displays the data. This approach used the “File” - “Import Text/PWL Data” menus in WaveView.

If the .PRINT TRAN line uses FORMAT=RAW, or the -r and -a command line options, to make a Xyce .raw file then the Xyce-generated binary and ASCII-formatted testTran.cir.raw files are also viewable in WaveView. In that case, the “File” - “Import Waveform File” menu options were used instead.

Probe format (FORMAT=CSD) may also work with WaveView for .PRINT TRAN output. However, that format will likely cause WaveView to crash for .PRINT AC or .PRINT DC. So, the use of Xyce-generated .csd files with WaveView is not recommended.
3.2. DC OUTPUT

The .DC netlist shown in Figure 3-2 produces a testDC.cir.prn file that is viewable in WaveView. In this example, the V(1) column will be used as the X-axis and the values of V(2) and V(3) will then be correctly plotted along that X-axis. This approach used the “File” - “Import Text/PWL Data” menus in WaveView.

If the .PRINT DC line uses FORMAT=RAW, or the -r and -a command line options, to make a Xyce .raw file then the Xyce-generated binary and ASCII-formatted testDC.cir.raw files are also viewable in WaveView. In that case, the “File” - “Import Waveform File” menu options were used instead.

3.3. AC OUTPUT

If the .PRINT AC line, for the netlist shown in Figure 3-3 uses FORMAT=RAW, or the -r and -a command line options, to make a Xyce .raw file then the Xyce-generated binary and ASCII-formatted testAC.cir.raw files are viewable in WaveView. In this case, the “File” -
“Import Waveform File” menus were used. Although, the default view in WaveView appears to be VM(b) and VM(c).

<table>
<thead>
<tr>
<th>Trivial high-pass filter circuit, implemented in Xyce</th>
</tr>
</thead>
<tbody>
<tr>
<td>****************************************************</td>
</tr>
<tr>
<td>R1 b c 1</td>
</tr>
<tr>
<td>R2 c 0 1</td>
</tr>
<tr>
<td>C1 a b 1u</td>
</tr>
<tr>
<td>V1 a 0 DC 0V AC 1</td>
</tr>
<tr>
<td>.print AC FORMAT=RAW v(b) v(c)</td>
</tr>
<tr>
<td>.ac dec 5 100Hz 1e6</td>
</tr>
<tr>
<td>.end</td>
</tr>
</tbody>
</table>

**Figure 3-3. Example Netlist for Xyce .AC Analysis**

If FORMAT=NOINDEX is used on the .PRINT AC line instead, then the output file’s header lines will need to be manually edited to match the examples given in the “Tabulated Data in Complex Numbers” section of the WaveView manual. That would change the first two lines of the testAC.cir.FD.prn file to match this format, where “xvar” is the frequency and “signal1” is v(b).

```
#format complex-ri
xvar signal1 signal2 ....
valuex value1-real value1-imag value2-real value2-imag ...  
```

There is also the HSPICE magnitude-phase format, which outputs the phase between 0 and 2*PI. The header lines for that format would need to be:

```
#format complex-mp
xvar signal1 signal2 ....
valuex value1-mag value1-phase value2-mag value2-phase ...  
```

Finally, there is also the HSPICE magnitude-angle format, which outputs the angle between 0 and 360 degrees. However, the Xyce output for the VP() operator is in radians, not degrees. So, that format would require additional post-processing of the Xyce output file.
4. HSPICE COMPATIBILITY IMPROVEMENTS IN XYCE 6.11

This chapter discuss some improvements to HSPICE compatibility that are included in the Xyce 6.11 release.

4.1. SUPPORT FOR TERNARY OPERATORS IN EXPRESSIONS

Support for ternary operators has been added, since that feature was a major barrier to using various foundry Process Design Kits (PDKs). So, an expression like this is now legal:

\[ V = \{ V(1) > 0 \ ? \ 2 \times V(1) : 0 \} \]

However, due to restrictions on the Xyce expression parser, it is essential that ternary operators never be written so that a bare parameter is directly to the left of a colon. This is because, in Xyce, colons are actually legal characters in parameters — the colon represents hierarchy, so that \( R1:R \) means the R parameter of device R1, and \( X1:A \) refers to the node A of subcircuit X1. Therefore, it is necessary to put at least one character that is invalid in parameter names in between the colon and the parameter. It is sufficient to use a space.

4.2. NESTED DELIMITERS IN EXPRESSIONS

Nested curly brackets are now legal in a Xyce expression. So, this set of netlist lines is now legal in Xyce expressions:

```
.param a1 = 2
.param a2 = \{ 1 + \{2*a1\} \}
```
4.3. DEVICE MODEL OPTIMIZATION

Several of the devices that had been generated from Verilog-A sources have been regenerated in such a manner as to realize a significant performance improvement over the versions in Xyce 6.10. The models that have been regenerated for this release are:

- VBIC 1.3 4-terminal (BJT level 12)
- HICUM levels 0 and 2 (BJT levels 230 and 234)
- PSP version 102 and 103 (MOSFET levels 102 and 103)
- JUNCAP 200 (Diode level 200)
- MEXTRAM (BJT level 504)
- MVS 2.0.0 (MOSFET levels 2000 (ETSOI) and 2001 (HEMT))

Performance improvements of up to a factor of four in the evaluation of model equations have been observed in these devices as a result of this change.

Furthermore, the VBIC 1.3 3-terminal model, which had previously had its derivatives optimized by hand in Xyce 6.10 (with a resulting performance improvement of approximately a factor of five) has also been regenerated in this manner, which has realized a further improvement in device evaluation of approximately 25% over Xyce 6.10.

4.4. CHARGE-BASED (OR CHARGE-CONSERVED) CAPACITOR MODEL

A new “charge expression” variant of the capacitor has been implemented. Users may now specify a solution-dependent expression for the capacitor’s “Q” parameter, and that expression will be used to evaluate the capacitor’s charge instead of by computing it from the capacitance. So, both of these instance lines are now legal in Xyce, where “expression” denotes a legal expression:

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

4.5. PATTERN (PAT) SOURCE FUNCTION

A pattern (PAT) source function was added for the V and I devices. So, for example, these instance line are now legal in Xyce:

```
V1 1 0 pat (5 0 0n 1n 2n 5n B1010 R=1)
I2 2 0 PAT 5 0 0n 1n 2n 5n B1010 R=1
```
The HSPICE parameters $RB$, $ENCODE$ and $RD\_INIT$, for the pattern source, are not supported. In addition, only the “0” and “1” states are supported in Xyce.

### 4.6. SUPPORT VOL AND CUR KEYWORDS FOR E AND G SOURCES

In HSPICE, $VOL$ is an allowed synonym for $VALUE$ for the $E$ source and $CUR$ is an allowed synonym for $VALUE$ for the $G$ source. Those synonyms are now supported Xyce 6.11, and all of these instance lines are now legal:

```
E1 1 2 VOL='V(9)'
G1 3 4 CUR='V(9)'
E2 5 6 VALUE=V(9)
G2 7 8 VALUE=V(9)
```

### 4.7. ALLOW X (AND X) AS SCALING FACTORS MEANING 1E6

Xyce now allows the use of “x” and “X” as scaling factors that mean 1e6. So, this instance line is now correct:

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

### 4.8. USE OF ARITHMETIC OPERATORS IN NODE AND DEVICE NAMES IN EXPRESSIONS

Something like this $G$-source instance line would produce a parsing error in Xyce 6.10:

```
G1 IOUT+ IOUT- VALUE={V(VIN+)}
```

The issue was that “arithmetic operators”, like $+$, were not valid in device and node names in as many expression contexts as in other Spice-like simulators. They are now allowed in device and node names in expressions, when they are enclosed within Xyce operators such as $V()$ and $I()$. See the “Legal Characters in Node and Device Names” subsection of the Xyce Reference Guide [1] for more details.
4.9. **ONGOING IMPROVEMENTS TO .MEASURE**

There are ongoing improvements to .MEASURE support in Xyce. For Release 6.11, support for MAX, MIN, PP and EQN measure modes for .AC analyses was added. Improved support for the PARAM measure and PAR() expression syntaxes was also added.

4.10. **PORT DEVICE**

The P-device (port device), which is used for S-Parameter analyses, is now supported in Xyce.

4.11. **-HSPICE-EXT COMMAND LINE OPTION**

For Release 6.11, a new command line option -hspice-ext has been added as an experimental feature. It allows the Xyce parser to accept a limited set of HSPICE syntaxes, in lieu of the Xyce ones. See Section [5.2](#) for more details.

Since this is an experimental option, DARPA POSH users should consult with Sandia about the best open-source Xyce version to use with this option. Improvements to, and bug fixes for, this feature may be available at https://github.com/Xyce.
5. STRATEGIES FOR TRANSLATING FROM HSPICE TO XYCE

Our long-term approach for translating HSPICE netlists into Xyce netlists, for our internal customers, is the Xyce Data Model (XDM) tool developed here at Sandia. XDM is already used internally at Sandia to translate between PSpice and Xyce netlists. However, it is not available for use yet by the DARPA POSH project for two reasons. First, the HSPICE-aware version of XDM is still under development. Second, the XDM tool has not been approved for Public Release yet.

In the meantime, this chapter provides an overview of other interim strategies for translating HSPICE netlists to run in Xyce where our internal approach uses a mixture of all three strategies. These approaches cover many of the issues for typical model libraries contained in Foundry Process Design Kits (PDKs). Finally, an updated version of this information may be found at the Xyce website.

5.1. MANUAL EDITING OF FILES

This list covers the most typical changes that are needed. In some foundry PDKs, the issues in Sections 2.1.6 and 2.1.9 have also been seen.

1. Open the file and remove any leading whitespace from lines in the file. (See Section 2.1.1)
2. Replace log with ln in expressions. (See Section 2.4.11)
3. Run the file using Xyce with a test netlist. If it runs without errors then “Yay!” More likely, additional steps will be needed.
4. Replace inline comments, per Section 2.1.2
5. Replace continuation characters outside of expressions. In this case, \\ is removed from the end of line, and + is added to the start of next line. (See Section 2.1.5)
6. Replace continuation characters from inside of expressions. \\ is removed from the end of line, and + is added to the start of next line. (See Section 2.1.13)
7. Replace logicals, per Section 2.4.11 So || becomes | and && becomes &.
8. Fix the exponentiation operator in expressions, per Section 2.4.11 So ^ becomes **.
9. Modify AGAUSS\((X,Y,X)\), in a “reversible way” that can be undone once AGUASS works in Xyce. (Note: AUNIF() functions will also need to be fixed.) One recommendation is that AGAUSS\((X,Y,X)\) becomes \(X\ ;\ \text{AGAUSS}(X,Y,Z)\). This replaces the AGUASS function with its mean \((X)\), and leaves the original statement as an inline comment.

10. .PARAM lines that defines functions will have to be replaced with .FUNC lines in Xyce. (See Section 2.1.14 and especially the discussion of how to handle HSPICE .PARAM lines that define multiple parameters and functions.)

11. Ternary expressions, mostly on .PARAM lines, may need to be modified to honor the caveat on spacing before the colon (:) operator discussed in Section 4.1.

12. On .LIB and .INC lines, single quotes should be replaced with double quotes, or simply removed.

13. For .PARAM lines that have Xyce “specials” in them, replace the name. The variable \(VT\) will have to be changed to something like \(VT1\). The HSPICE special TEMPER would be changed to the corresponding Xyce special, which is TEMP. (See Section 2.1.15.)

14. Change something like .TEMP 25 to the Xyce equivalent of .STEP TEMP 25 25 1. This approach is currently preferred to the use of .OPTIONS DEVICE TEMP=25.

15. Account for any differences in TNOM between HSPICE and Xyce. (See Section 2.4.17.)

16. Any usages of DTMP for subcircuits will likely need to be replaced with explicit parameter values, especially for DTMP not equal to 0.

17. Any HSPICE .OPTION lines should likely be commented out.

18. Any HSPICE .TRAN lines may need to be converted to Xyce format, per Section 2.4.13.

19. Fix independent and controlled source instance lines, as needed, per Sections 2.1.18 and 2.1.17.

5.2. -HSPICE-EXT COMMAND LINE OPTION

For Release 6.11, a new command line option -hspice-ext has been added as an experimental feature. It addresses some of the issues (2, 7, 8 and 9) discussed in the previous section, that were deemed either “easy to toggle on/off” or “really annoying to fix in a text editor”. It tells the Xyce parser to accept the HSPICE syntaxes rather than the Xyce syntaxes for the limited set of cases shown in Table 5-1. Unsupported, or blank, names (e.g., bogo) are a Xyce parsing error.
### Table 5-1. `-hspice-ext` Command Line Option

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-hspice-ext units</code></td>
<td>Toggles on ( a=1e-18 ) as a scaling factor</td>
</tr>
<tr>
<td><code>-hspice-ext math</code></td>
<td>Toggles in the HSPICE meanings for the logical operators (</td>
</tr>
<tr>
<td><code>-hspice-ext random</code></td>
<td>AGAUSS() and GAUSS() will return the mean value rather than a random variate</td>
</tr>
<tr>
<td><code>-hspice-ext all</code></td>
<td>Does all three</td>
</tr>
<tr>
<td><code>-hspice-ext units,math,random</code></td>
<td>A comma-separated listing is also legal. This example is equivalent to <code>all</code>.</td>
</tr>
</tbody>
</table>

Since this is an experimental option, DARPA POSH users should consult with Sandia about the best open-source Xyce version to use with this option. Improvements to, and bug fixes for, this feature may be available at https://github.com/Xyce.

### 5.3. SCRIPT-BASED APPROACH

Sandia also has a (likely under-tested) Matlab script that is used internally to address many of the issues discussed in Section 5.1. It is known to work with Matlab Release 2018b, but should work with earlier versions also. It has not been tested with Octave. This Matlab script is not currently approved for public release, but it could be if there was sufficient interest.
REFERENCES


## DISTRIBUTION

Email—Internal (encrypt for OUO)

<table>
<thead>
<tr>
<th>Name</th>
<th>Org.</th>
<th>Sandia Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical Library</td>
<td>01177</td>
<td><a href="mailto:libref@sandia.gov">libref@sandia.gov</a></td>
</tr>
</tbody>
</table>