SAPICE = SAGE + NGSPICE

Symbolic Analysis from sPICE netlists in SAgemath

Alessandro Bernardini

January 25, 2014

Version: 0.3.

Author: Alessandro Bernardini

alessandro.bernardini.tum@gmail.com

In the present version the program was NOT well tested, so bugs are expected: please report all kind of bugs and isses you will find.

Project home: https://github.com/alessandro-bernardini/SAPICE

Requires: ngspice revision 24; sage version 5.6. Other version of ngspice or sage should work well too.

`http://www.sagemath.org/`
`http://ngspice.sourceforge.net/download.html`
Date: June 2013

1 Overview

Note: consider the generated doxygen documentation also in the project home. See:
`https://github.com/alessandro-bernardini/SAPICE`
`http://alessandro-bernardini.github.io/SAPICE/`
(in the corresponding subdirectories)

The program SAPICE for now consists of the python module `sage_circuit_analysis.py`
This python package is intended for use in SAGE. ngspice is required too.
It provides the class `SmallSignalLinearCircuit` which reads an NGSPICE netlist and computes in symbolic form the nodal equations for the linearized small signal circuit in the Laplace domain. Those nodal equations in the Laplace domain can be fully symbolic or the numerical values read from the NGSPICE netlist can be substituted for the circuit parameters.
The so obtained equations can be solved with SAGE.

Project homepage: `https://github.com/alessandro-bernardini/SAPICE`
Works with: ngspice revision 24; sage version 5.6
Download link for SAGE:
`http://www.sagemath.org/`
Download link for NGSPICE:
`http://ngspice.sourceforge.net/download.html`
or via `sudo apt-get install ngspice`

2 Usage

Start with a description of your circuit's netlist in an ngspice circuit file `circuitfile.cir`.

Only resistor, capacitors, inductors, current sources, voltage controlled current sources and BJTs are supported for now. *The netlist must be flat.*
Independent voltage sources are not supported and an independent voltage sources has to be replaced with an electrically equivalent current source (source transformation). In doing so the internal resistance of the source has to be considered.
This is because for now only nodal analysis is provided. In future versions modified nodal analysis is planned and this will support voltage sources and short circuits.
A short circuit ($R = 0\,\Omega$) is also not supported for now and for dealing with a short circuit the original ngspice netlist has to be modified, identifying both

nodes of the short with the same node identifier in the netlist. In alternative a sufficiently low resistance can approximate the short.

The ground node must be named 0 or *gnd* or *GND*.

The netlist must be processed by ngspice and a batch output file `batchoutput` is required for the *.OP* data and the description of BJT models (so put an *.OP* line in your ngspice file - no other analyses should be activated except the operating point computation with *.OP*).

For this run the command (from the unix prompt):

```
ngspice -b circuitfile.cir -o batchoutput.log
```

Then start SAGE.

From the sage prompt

```
import sage_circuit_analysis as sca
```

assuming that the file is found in the current directory.

Now you can use the classes provided in the module.

For example: from SAGE prompt:

```
 circuit = sca.SmallSignalLinearCircuit(
  "examples/colpitts/colpitts_bjtmodel.cir",
"examples/colpitts/colpitts_bjtmodel.log")
```

`circuit` will be an object that symbolically describes the circuit in SAGE (write all in one line in the sage prompt).

(The log file must have been generated previously as described above)

You can provide additional options in the constructor of the: `check_operating_region=False` will assume that the transistors are in the forward active region of operation without checking if this is true; `set_default_ic_to_zero=False` will not consider a default zero initial condition when initial conditions are *not* provided; `ignore_all_ic=True` will set to zero every initial condition, even those explicitly specified and this both for symbolic nodal equations and for nodal equations with substituted numerical values regarding the circuit parameters.

3 Description of methods and data members

See the provided doxygen documentation and the docstrings.

We will describe the main methods and data members of the class SmallSignalLinearCircuit.

`.nodal_equations`

prints the nodal equations for the small signal linearized circuit in the Laplace domain as a dictionary that associates to each node the relative nodal equation in symbolic form in SAGE

`.nodal_equations_substitutions`

prints the nodal equations for the small signal linearized circuit in the Laplace domain as a list for SAGE, substituitng the numerical values given in the netlist for the circuit parameters. The indipendent current sources are always in symbolic form a function of the variable s.

`.print_information()`
prints (not still complete) informations about the circuit and the equations.

`.solve_nodal_equations_symb()`
solves the nodal equations for nodal analysis of the small signal linearized circuit in the Laplace domain in symbolic form. The computation can take time. The result is a list containing a single element which is a list which contains the nodal voltages in symbolic form[1].

`.solve_nodal_equations_num()`
solves the nodal equations for the nodal analysis of the small signal linearized circuit in the Laplace domain replacing the numerical values for the circuit parameters. The type of the result is as for the symbolic case. The nodal voltages are now given as polynoms in s with numeric coefficients.

`.additional_equations`
prints additional equations linking the linearized small signal circuit quantities to operating pint values and other parameters.

`.additional_equation_explicit`
prints additional equations linking the linearized small signal circuit quantities to operating point values and other parameters in the form of a dictionary that can be used for substitutions in SAGE.

`.prin_lin_circuit`
prints the linearized small signal linear circuit. Usefull for further manipulation of the circuit. The output can then be feed back to SAPICE.

`.default_substitutions`
prints the default substitutions as a dictionary. Usefull for substitutions in SAGE.

`.default_substitutions_values`
prints the default substitutions as a dictionary. Usefull for substitutions in SAGE. Numerical values are considered for the small signal equivalent circuit elements too that depends on the operating point of semiconductor devices.

`.initial_conditions`
retuns the initial conditions for (small signal equivalent) circuit elements.

`.nodal_voltages`
retuns the nodal voltages as a list.

`.print_symbols()`

[1] more precisely the nodal voltages are given as sage expressions in the form $Vnodename == expression$.

prints an (incomplete) explanation of used symbols.

4 Some typical uses

4.1 Computation of poles and zeros

Describe the circuit in a ngspice netlist (in a .cir file) including small signal independent current sources when computing transfer functions. If the DC current of the indipendent source has to be zero, then use a description of the form

$$\mathtt{I}n \quad n+ \quad n- \quad 0$$

Follow the steps previously described in this documentation.

Once the `circuitinstance` is created (as previously described) do:

```
NODALVOLTAGES = circuit.solve_nodal_equations_symb()
NODALVOLTAGESNUM = circuit.solve_nodal_equations_num()
```

With

```
NODALVOLTAGES[0][0]
NODALVOLTAGES[0][1]
...
NODALVOLTAGES[0][n]

NODALVOLTAGESNUM[0][0]
NODALVOLTAGESNUM[0][1]
...
NODALVOLTAGESNUM[0][n]
```

you obtain the expressions for the nodal voltages in symbolic and numerical form. `n=`*number_of_nodes* (different from ground).

For the numerical computation of the poles do (always in the sage prompt):

```
s = var('s')
R = RR[s]
R(NODALVOLTAGESNUM[0][0].rhs().denominator()).roots(CDF)
```

Instead of NODEVOLTAGESNUM[0][0] try different nodes also NODEVOLTAGENUM[0][1], etc...

This will return the complex roots of the denominator of the expression for the voltage `NODALVOLTAGESNUM[0][i].lhs()` for the considered node i (in the example $i = 0$).

See the sagemath documentation for details.

Also see the doctstrings and/or the doxygen documentation in the project directory.

5 Known issues

The software is only in the initial stage and was NOT really tested.

Please contribute in testing the software and report bugs and/or writing a better documentation.