

SAPICE = SAGE + NGSPICE

Generated by Doxygen 1.8.1.2

Sat Jan 25 2014 17:21:07

Contents

1	Main Page	1
2	Namespace Index	3
2.1	Packages	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	sage_circuit_analysis Namespace Reference	9
5.1.1	Function Documentation	9
5.1.1.1	extract_value	9
5.1.1.2	get_inductor_data	10
5.1.1.3	rational_func	10
5.1.1.4	simplify_polynomial	10
5.1.1.5	simplify_rational_func	10
5.1.1.6	simplify_sum	10
5.1.2	Variable Documentation	10
5.1.2.1	BJT_EXPR	10
5.1.2.2	CAPACITOR_EXPR	10
5.1.2.3	DATA_FIELD	11
5.1.2.4	I_EXPR	11
5.1.2.5	INDUCTOR_EXPR	11
5.1.2.6	KIND_EXPR	11
5.1.2.7	RESISTOR_EXPR	11
5.1.2.8	TEMP_EXPR	11
5.1.2.9	V_EXPR	11
5.1.2.10	VCCS_EXPR	11
6	Class Documentation	13

6.1	sage_circuit_analysis.SmallSignalLinearCircuit Class Reference	13
6.1.1	Detailed Description	15
6.1.2	Constructor & Destructor Documentation	15
6.1.2.1	__init__	16
6.1.3	Member Function Documentation	16
6.1.3.1	clone	16
6.1.3.2	dict_default_vals	16
6.1.3.3	export_lin_circuit_graph	16
6.1.3.4	export_lin_circuit_string	17
6.1.3.5	get_lin_circuit_as_string	17
6.1.3.6	impedance	17
6.1.3.7	print_information	17
6.1.3.8	print_lin_circuit	18
6.1.3.9	print_symbols	18
6.1.3.10	solve_nodal_equations_num	18
6.1.3.11	solve_nodal_equations_symb	18
6.1.3.12	transimpedance	18
6.1.3.13	write_circuit_graph_dot	19
6.1.3.14	write_lin_circuit_to_file	19
6.1.3.15	z_parameters	19
6.1.4	Member Data Documentation	19
6.1.4.1	add_eq_sol	19
6.1.4.2	bjt_id	19
6.1.4.3	bjt_line_data	19
6.1.4.4	bjt_lineptr	19
6.1.4.5	bjt_model_desc_reached	19
6.1.4.6	bjt_reached	19
6.1.4.7	cap_line_data	19
6.1.4.8	cap_lineptr	19
6.1.4.9	capacitor_id	19
6.1.4.10	circuit_file	19
6.1.4.11	coupled_to	19
6.1.4.12	cpld_inductor_data	20
6.1.4.13	cpld_inductor_node	20
6.1.4.14	cpld_voltage_subst	20
6.1.4.15	data_bjt	20
6.1.4.16	data_bjt_strip	20
6.1.4.17	data_cap	20
6.1.4.18	data_cap_strip	20
6.1.4.19	data_i	20

6.1.4.20	data_i_strip	20
6.1.4.21	data_ind	20
6.1.4.22	data_ind_strip	20
6.1.4.23	data_kind	20
6.1.4.24	data_kind_strip	20
6.1.4.25	data_res	20
6.1.4.26	data_res_strip	20
6.1.4.27	data_temp	20
6.1.4.28	data_vccs	20
6.1.4.29	data_vccs_strip	20
6.1.4.30	data_vccs_strip	20
6.1.4.31	flags	20
6.1.4.32	i_id	20
6.1.4.33	i_line_data	20
6.1.4.34	i_lineptr	20
6.1.4.35	id	20
6.1.4.36	id	20
6.1.4.37	ignore_all_ic	21
6.1.4.38	ind_id	21
6.1.4.39	ind_line_data	21
6.1.4.40	ind_lineptr	21
6.1.4.41	inductor_id	21
6.1.4.42	inductor_node0	21
6.1.4.43	inductor_node1	21
6.1.4.44	K_coeff	21
6.1.4.45	K_coeff	21
6.1.4.46	kind_id	21
6.1.4.47	kind_line_data	21
6.1.4.48	kind_lineptr	21
6.1.4.49	matchline	21
6.1.4.50	model	21
6.1.4.51	model_present_bjt_reached	21
6.1.4.52	ngspice_line_data	21
6.1.4.53	operating_region	21
6.1.4.54	original_circuit_file	21
6.1.4.55	pos	21
6.1.4.56	present_bjt_reached	21
6.1.4.57	res_line_data	21
6.1.4.58	res_lineptr	21
6.1.4.59	resistor_id	21

6.1.4.60	set_default_ic_to_zero	21
6.1.4.61	simpl_temp	21
6.1.4.62	skipnextline	21
6.1.4.63	spice_batch_output_file	21
6.1.4.64	subs_zero_ic	22
6.1.4.65	temp_dict	22
6.1.4.66	temperature	22
6.1.4.67	type	22
6.1.4.68	value	22
6.1.4.69	vccs_id	22
6.1.4.70	vccs_line_data	22
6.1.4.71	vccs_lineptr	22
6.2	sage_circuit_analysis.WrongData Class Reference	22
6.2.1	Constructor & Destructor Documentation	22
6.2.1.1	__init__	22
6.2.2	Member Data Documentation	22
6.2.2.1	data	22
6.3	sage_circuit_analysis.WrongUse Class Reference	22
6.3.1	Constructor & Destructor Documentation	23
6.3.1.1	__init__	23
6.3.2	Member Data Documentation	23
6.3.2.1	data	23
7	File Documentation	25
7.1	sage_circuit_analysis.py File Reference	25

Chapter 1

Main Page

Author: Alessandro Bernardini

`alessandro.bernardini@tum.de`

<https://github.com/alessandro-bernardini/SAPICE>

<http://alessandro-bernardini.github.io/SAPICE/>

License: GNU GPL

Disclaimer: THERE IS NO WARRANTY FOR THE PROGRAM (SAPICE and all its provided components), TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS (Alessandro Bernardini) AND/OR OTHER PARTIES PROVIDE THE PROGRAM AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THIS PROGRAM WAS NOT WELL TESTED !

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>

See documentation and license

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

sage_circuit_analysis	9
-------------------------------------------------	---

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

sage_circuit_analysis.SmallSignalLinearCircuit	13
sage_circuit_analysis.WrongData	22
sage_circuit_analysis.WrongUse	22

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

sage_circuit_analysis.py	25
----------------------------------------------------	----

Chapter 5

Namespace Documentation

5.1 sage_circuit_analysis Namespace Reference

Classes

- class [SmallSignalLinearCircuit](#)
- class [WrongUse](#)
- class [WrongData](#)

Functions

- def [extract_value](#)
- def [simplify_sum](#)
- def [simplify_polynomial](#)
- def [simplify_rational_func](#)
- def [rational_func](#)
- def [get_inductor_data](#)

Variables

- tuple [RESISTOR_EXPR](#) = re.compile('^(\\s)*(R|r)\\w+')
- tuple [CAPACITOR_EXPR](#) = re.compile('^(\\s)*(C|c)\\w+')
- tuple [INDUCTOR_EXPR](#) = re.compile('^(\\s)*(L|l)\\w+')
- tuple [V_EXPR](#) = re.compile('^(\\s)*(V|v)\\w+')
- tuple [I_EXPR](#) = re.compile('^(\\s)*(I|i)\\w+')
- tuple [VCCS_EXPR](#) = re.compile('^(\\s)*(G|g)\\w+')
- tuple [BJT_EXPR](#) = re.compile('^(\\s)*(Q|q)\\w+')
- tuple [KIND_EXPR](#) = re.compile('^(\\s)*(K|k)\\w+')
- tuple [DATA_FIELD](#) = re.compile("\\s+\\w+\\=[-\\+]?\\w*\\.\\.?\\w+\\[-\\+]?\\d*|\\s+\\[-\\+]?\\w*\\.\\.?\\w+\\[-\\+]?\\d*|\\s+\\{\\w+\\}")
- tuple [TEMP_EXPR](#) = re.compile('^(\\s)*(\\.temp)\\s+\\w+', re.IGNORECASE)

5.1.1 Function Documentation

5.1.1.1 def sage_circuit_analysis.extract_value (data)

extracts numerical value from a data field. For example 0.003 from 3m. data must be a stripped string

5.1.1.2 `def sage_circuit_analysis.get_inductor_data (line, ignore_all_ic, set_default_ic_to_zero)`

5.1.1.3 `def sage_circuit_analysis.rational_func (large_expression)`

5.1.1.4 `def sage_circuit_analysis.simplify_polynomial (polinom, dictionary, treshhold = 0, variable = sage.var('s'), safe_check = True)`

Returns a tuple (result, polinom-result)
where result is an approximation of the polinom polinom given as a sage expression.
The polynom is considered in the variable variable
Default: variable=sage.var('s')

dictionary is a dictionary that provides numerical values for the symbolic variables
in the coefficeints of the polinom.

treshhold is a number in [0,1[that fixes a treshhold: for treshhold=0 result will be equal
to polinom.

Every coefficient of the polynomial (coefficient made by a sum of terms) will be simplified
invoking simplify_sum.

5.1.1.5 `def sage_circuit_analysis.simplify_rational_func (rational_func, dictionary, treshhold = 0, variable = sage.var('s'), safe_check = True)`

simplifies a rational function rational_func (a sage expression)
in the variable variable (default sage.var('s')).

dictionary is a dictionary that provides numerical values for the symbolic variables
in the coefficeints of the polinom.

treshhold is a number in [0,1[that fixes a treshhold: for treshhold=0 result will be equal
to polinom.

A simplified numerator and denominator will be computed invoking simplify_polynomial

The result will be a dictionary
(result, rational_func - result)
with result a simplified version of rational_func

5.1.1.6 `def sage_circuit_analysis.simplify_sum (expr, dictionary, treshhold = 0)`

Returns a tuple (simplified_expr, expr-simplified_expr)
where simplified_expr is an approximation of expr.

expr is a symbolic sage expression made of terms added together (a sum of terms)

dictionary is a dictionary that provides numerical values for the symbolic variables
in the expression expr

treshhold is a number in [0,1[that fixes a treshhold.

If the numerical value of a given term is lower than the numerical value of the term
that evaluates to the maximum (along all terms) multiplied with the trashhold then the
corresponding term is ignored.

5.1.2 Variable Documentation

5.1.2.1 `tuple sage_circuit_analysis.BJT_EXPR = re.compile('^(\s)*(Q|q)\w+')`

5.1.2.2 `tuple sage_circuit_analysis.CAPACITOR_EXPR = re.compile('^(\s)*(C|c)\w+')`

5.1.2.3 tuple sage_circuit_analysis.DATA_FIELD = re.compile('s+ w+ \=[-+]? w* \. ? w+ [-+]? d* | s+ [-+]? w* \. ? w+ [-+]? d* | s+ { w+ }')

5.1.2.4 tuple sage_circuit_analysis.I_EXPR = re.compile('^(s)*(l|i) w+')

5.1.2.5 tuple sage_circuit_analysis.INDUCTOR_EXPR = re.compile('^(s)*(L|l) w+')

5.1.2.6 tuple sage_circuit_analysis.KIND_EXPR = re.compile('^(s)*(K|k) w+')

5.1.2.7 tuple sage_circuit_analysis.RESISTOR_EXPR = re.compile('^(s)*(R|r) w+')

5.1.2.8 tuple sage_circuit_analysis.TEMP_EXPR = re.compile('^(s)*(.temp) s+ w+', re.IGNORECASE)

5.1.2.9 tuple sage_circuit_analysis.V_EXPR = re.compile('^(s)*(V|v) w+')

5.1.2.10 tuple sage_circuit_analysis.VCCS_EXPR = re.compile('^(s)*(G|g) w+')

Chapter 6

Class Documentation

6.1 sage_circuit_analysis.SmallSignalLinearCircuit Class Reference

Public Member Functions

- def [__init__](#)
- def [clone](#)
- def [solve_nodal_equations_symb](#)
- def [solve_nodal_equations_num](#)
- def [impedance](#)
- def [transimpedance](#)
- def [z_parameters](#)
- def [dict_default_vals](#)
- def [print_information](#)
- def [print_lin_circuit](#)
- def [get_lin_circuit_as_string](#)
- def [write_lin_circuit_to_file](#)
- def [export_lin_circuit_graph](#)
- def [export_lin_circuit_string](#)
- def [print_symbols](#)
- def [write_circuit_graph_dot](#)

Public Attributes

- [circuit_file](#)
- [original_circuit_file](#)
- [spice_batch_output_file](#)

Static Public Attributes

- int [pos](#) = 0
- tuple [data_temp](#) = DATA_FIELD.findall(line)
- list [temperature](#) = [data_temp](#)[0]
- tuple [bjt_lineptr](#) = BJT_EXPR.match(line)
- list [bjt_line_data](#) = line[bjt_lineptr.end():]
- list [bjt_id](#) = line[:bjt_lineptr.end()]
- string [operating_region](#) = 'unknown'
- tuple [data_bjt](#) = DATA_FIELD.findall([bjt_line_data](#))
- list [data_bjt_strip](#) = []

- list `model` = `data_bjt_strip`[3]
- `bjt_model_desc_reached` = False
- `model_present_bjt_reached` = False
- `skipnextline` = False
- `flags` = `re.IGNORECASE`!None:
- `matchline` = \
- `bjt_reached` = False
- `present_bjt_reached` = False
- `add_eq_sol` = \
- tuple `kind_lineptr` = `KIND_EXPR.match(line)`
- list `kind_line_data` = `line[kind_lineptr.end():]`
- list `kind_id` = `line[:kind_lineptr.end()]`
- tuple `data_kind` = `DATA_FIELD.findall(kind_line_data)`
- list `data_kind_strip` = []
- tuple `ind_lineptr` = `INDUCTOR_EXPR.match(line2)`
- list `ind_id` = `line2[:ind_lineptr.end()]`
- `ignore_all_ic` = `ignore_all_ic`,
- `set_default_ic_to_zero` = `set_default_ic_to_zero`)
- string `type` = 'L'
- dictionary `coupled_to`
- list `value` = `self.coupled_inductors_data[ind_id]`
- list `cpld_inductor_data` = `self.coupled_inductors_data[cpld_inductor]`
- dictionary `cpld_inductor_node` = {}
- int `K_coeff` = 0
- list `inductor_node0` = `self.coupled_inductors_data[inductor]`
- list `inductor_node1` = `self.coupled_inductors_data[inductor]`
- tuple `K_coeff` = `sage.var(self.coupled_inductors_matrix[cpld_inductor][inductor])`
- dictionary `temp_dict` = {}
- dictionary `cpld_voltage_subst` = {}
- list `simpl_temp` = []
- tuple `i_lineptr` = `I_EXPR.match(line)`
- list `i_line_data` = `line[i_lineptr.end():]`
- list `i_id` = `line[:i_lineptr.end()]`
- tuple `data_i` = `DATA_FIELD.findall(i_line_data)`
- list `data_i_strip` = []
- `id` = `i_id`,
- list `ngspice_line_data` = `data_i_strip[0:2]`
- tuple `res_lineptr` = `RESISTOR_EXPR.match(line)`
- list `res_line_data` = `line[res_lineptr.end():]`
- list `resistor_id` = `line[:res_lineptr.end()]`
- tuple `data_res` = `DATA_FIELD.findall(res_line_data)`
- list `data_res_strip` = []
- list `id` = `resistor_id`,`valuedata_res_strip[2:]`
- tuple `cap_lineptr` = `CAPACITOR_EXPR.match(line)`
- list `cap_line_data` = `line[cap_lineptr.end():]`
- list `capacitor_id` = `line[:cap_lineptr.end()]`
- tuple `data_cap` = `DATA_FIELD.findall(cap_line_data)`
- list `data_cap_strip` = []
- list `ind_line_data` = `line[ind_lineptr.end():]`
- list `inductor_id` = `line[:ind_lineptr.end()]`
- tuple `data_ind` = `DATA_FIELD.findall(ind_line_data)`
- list `data_ind_strip` = []
- tuple `vccs_lineptr` = `VCCS_EXPR.match(line)`
- list `vccs_line_data` = `line[vccs_lineptr.end():]`
- list `vccs_id` = `line[:vccs_lineptr.end()]`

- tuple `data_vccs` = `DATA_FIELD.findall(vccs_line_data)`
- list `data_vccs_strip` = []
- `data_vccs_strip` = `data_vccs_strip\`
- dictionary `subs_zero_ic` = {}

6.1.1 Detailed Description

PROJECT HOME:

```
https://github.com/alessandro-bernardini/SAPICE
http://alessandro-bernardini.github.io/SAPICE/
```

This class implements nodal analysis for linear electrical circuits. If a nonlinear circuit is given then the linearized small signal circuit is considered.

This class is designed to be used WITHIN the sage computer algebra system

```
http://www.sagemath.org/
```

and in conjunction with ngspice (open source version of spice):

```
http://ngspice.sourceforge.net/
```

So you have to install sage and ngspice first (see relative documentation).

Then run

```
sage
```

from the shell

then in the sage prompt import the present module

```
import sage_circuit_analysis
```

and then create a `SmallSignalLinearCircuit` object

```
circuit = sage_circuit_analysis.SmallSignalLinearCircuit('circuitfile.cir')
```

with `circuitfile.cir` a spice netlist (see additional information below and in the help of the constructor).

You can read-in a ngspice netlist and typically you will provide the operating point data and other information via an ngspice batch output file that you have to generate with the command (from the shell):

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

where `circuitfile.cir` is the spice netlist

and `batchoutput.log` is the log file containing the output data of interest.

Both files `circuitfile.cir` and `batchoutput.log` must be passed to a `SmallSignalLinearCircuit` object when invoking the constructor.

The nodal equations can be solved both numerically (if numerical values are provided) and symbolically.

You can compute impedances and two port network parameters and do the computation of poles and zeroes if a symbolic closed form solutions exists.

It is possible to compute symbolic approximations to the exact solution where only dominant terms are considered.

NOTE: numerical solutions usually means a polynomial or a rational function dependent on the complex variable s and on independent sources.

Usually all quantities are complex.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `def sage_circuit_analysis.SmallSignalLinearCircuit.__init__(self, filename = None, spice_batch_output_file = None, circuit_netlist = None, check_operating_region = True, set_default_ic_to_zero = True, ignore_all_ic = False)`

filename='circuitfile.cir'

where circuitfile.cir is a ngspice (spice) netlist.

spice_batch_output_file='batchoutput.log'

a file needed when operating point data must be considered

You have to run from the shell the command

ngspice -b circuitfile.cir -o batchoutput.log

for generating the batchoutput.log file

in alternative

circuit_netlist=CIRCUIT_STRING

can be used where CIRCUIT_STRING is a string containing

the circuit netlist (in place of circuitfile.cir)

with check_operating_region=True a linearized model in dependence of the operating region will be chosen for each semiconductor devices.

Otherwise it will be assumed that a default model is valid. (for

BJT transistors the active region small signal model will be this

default model)

ignore_all_IC=True will ignore all initial conditions for both symbolic and numeric computations

set_default_ic_to_zero=True will set initial conditions to zero for capacitors or inductors where no initial condition is specified

6.1.3 Member Function Documentation

6.1.3.1 `def sage_circuit_analysis.SmallSignalLinearCircuit.clone(self)`

returns a deepcopy of the object

6.1.3.2 `def sage_circuit_analysis.SmallSignalLinearCircuit.dict_default_vals(self)`

returns a pair of dictionaries (dict_num, dict_sym)

where

dict_num is a dictionary containing as keys the circuit parameters and as values the corresponding numerical values

dict_sym is a dictionary containing as keys the circuit parameters for which a numerical value was not explicitly determined and as values the symbolic expressions associated with the considered circuit parameter.

If all numerical values can be explicitly computed then

dict_sym will be empty.

6.1.3.3 `def sage_circuit_analysis.SmallSignalLinearCircuit.export_lin_circuit_graph(self, shorts = None, remove = None, with_substitutions = True)`

returns a graph (a networkx object) of the linear circuit

shorts is a dictionary {'node1':'node2', 'node3':'node4'}

where node1 will be shorted with node2

and node3 will be shorted with node 4

etc.

A key node should not also be a value node for preventing conflicts, so when shorting multiple nodes together the proper ordering is important.

remove is a list of components to remove (replace with open circuit)

For example remove=['R1','C1']

will remove the resistor R1 and the capacitor C1

from the netlist.

with_substitutions=True
will explicitly consider numerical values resulting from the operating point of nonlinear devices
with_substitutions=False will leave the values of parameters resulting from the operating point of nonlinear devices as a symbolic expression

6.1.3.4 `def sage_circuit_analysis.SmallSignalLinearCircuit.export_lin_circuit_string (self, shorts=None, remove=None, with_substitutions=True, write_to_file=None)`

returns a string describing the netlist of the linear(ized) circuit

shorts is a dictionary {'node1':'node2', 'node3':'node4'}
where node1 will be shorted with node2
and node3 will be shorted with node 4
etc.

A key node should not also be a value node for preventing conflicts, so when shorting multiple nodes together the proper ordering is important.

remove is a list of components to remove (replace with open circuit)
For example remove=['R1','C1']
will remove the resistor R1 and the capacitor C1
from the netlist.

with_substitutions=True
will explicitly consider numerical values resulting from the operating point of nonlinear devices
with_substitutions=False will leave the values of parameters resulting from the operating point of nonlinear devices as a symbolic expression

6.1.3.5 `def sage_circuit_analysis.SmallSignalLinearCircuit.get_lin_circuit_as_string (self)`

returns a string containing the linear circuit
(or linearized circuit) netlist

6.1.3.6 `def sage_circuit_analysis.SmallSignalLinearCircuit.impedance (self, port, with_substitutions=True, symbolic=True)`

computes the impedance between two nodes in the circuit

port=('node1','node2')
is the pair of nodes considered for the impedance computations, for example
port=('0','1')
will compute the impedance between node '1' (if a node '1' is given in the circuit) and ground (node '0').

with_substitutions=True
(default) will consider the actual values of all small signal circuit parameters that results from the consideration of the operating point of nonlinear devices. Otherwise those parameters will be left symbolic.
(affects the computation of numerical impedance values)

symbolic=True
impedance is computed symbolically. (symbolic=False will compute a numeric impedance value)

All internal sources and all the initial conditions will be ignored (set to zero appropriately).

See also the help for the transimpedance method.

6.1.3.7 `def sage_circuit_analysis.SmallSignalLinearCircuit.print_information (self)`

prints some information about the circuit

6.1.3.8 def sage_circuit_analysis.SmallSignalLinearCircuit.print_lin_circuit (self)

prints the linear (or linearized) circuit netlist

6.1.3.9 def sage_circuit_analysis.SmallSignalLinearCircuit.print_symbols (self)

displays information on the symbols adopted by the program (incomplete)

6.1.3.10 def sage_circuit_analysis.SmallSignalLinearCircuit.solve_nodal_equations_num (self, set_ind_ss_src_to_zero = False)

Returns a numeric solutions of the nodal equations.

The solution will be a list containing a list of equations that have on the left hand side the nodal voltages and on the right hand side the symbolic expression representing the solution.

set_ind_ss_src_to_zero=True will set all the independent small signal sources (in the small signal linearized circuit) to zero when computing the solution

6.1.3.11 def sage_circuit_analysis.SmallSignalLinearCircuit.solve_nodal_equations_symb (self, set_ind_ss_src_to_zero = False)

Returns a symbolic solutions of the nodal equations.

The solution will be a list containing a list of equations that have on the left hand side the nodal voltages and on the right hand side the symbolic expression representing the solution.

set_ind_ss_src_to_zero=True will set all the independent small signal sources (in the small signal linearized circuit) to zero when computing the solution

6.1.3.12 def sage_circuit_analysis.SmallSignalLinearCircuit.transimpedance (self, port_I_in, port_V_out, with_substitutions = True, symbolic = True)

computes the trans-impedance between two nodes in the circuit

port_I_in=('node1','node2')

is the pair of nodes considered for the current input
where the (positive) impressed input current flows into node2 (and out of node1)

port_V_out=('node3','node4')

is the pair of nodes considered for the output voltage:
output_voltage = Vnode4 - Vnode3

the transimpedance will then be the output voltage/input current

symbolic=True
computes a symbolic expression as solution

with_substitutions=True
consider the operating point data (affects numerical results with symbolic=False).

internal sources or initial conditions are ignored.

6.1.3.13 `def sage_circuit_analysis.SmallSignalLinearCircuit.write_circuit_graph_dot (self, output_graph_file = 'circuit_graph.dot', with_substitutions = True, edge_labels = True)`

generates a dot file in which a representation of the circuit graph is given.

output_graph_file='circuit_graph.dot'

file in which the graph representation will be written.

with_substitutions=True

will consider numerical values resulting from the operating point of nonlinear devices

edge_labels=True

will labels edge with informations.

6.1.3.14 `def sage_circuit_analysis.SmallSignalLinearCircuit.write_lin_circuit_to_file (self, filename)`

writes the linear circuit netlist to file 'filename'

6.1.3.15 `def sage_circuit_analysis.SmallSignalLinearCircuit.z_parameters (self, port_in, port_out, with_substitutions = True, symbolic = True)`

returns a sage matrix containing the two port network z-parameters

port_in=('node1', 'node2')

port_out=('node3', 'node4')

where nodei is a node identifier

for any port

port_x=('nodea', 'nodeb')

the positive port voltages are nodeb - nodea

and the positive currents are flowing into nodeb (and out of nodea)

6.1.4 Member Data Documentation

6.1.4.1 `sage_circuit_analysis.SmallSignalLinearCircuit.add_eq_sol = \` [static]

6.1.4.2 `list sage_circuit_analysis.SmallSignalLinearCircuit.bjt_id = line[:bjt_lineptr.end()]` [static]

6.1.4.3 `list sage_circuit_analysis.SmallSignalLinearCircuit.bjt_line_data = line[bjt_lineptr.end():]` [static]

6.1.4.4 `tuple sage_circuit_analysis.SmallSignalLinearCircuit.bjt_lineptr = BJT_EXPR.match(line)` [static]

6.1.4.5 `sage_circuit_analysis.SmallSignalLinearCircuit.bjt_model_desc_reached = False` [static]

6.1.4.6 `sage_circuit_analysis.SmallSignalLinearCircuit.bjt_reached = False` [static]

6.1.4.7 `list sage_circuit_analysis.SmallSignalLinearCircuit.cap_line_data = line[cap_lineptr.end():]` [static]

6.1.4.8 `tuple sage_circuit_analysis.SmallSignalLinearCircuit.cap_lineptr = CAPACITOR_EXPR.match(line)` [static]

6.1.4.9 `list sage_circuit_analysis.SmallSignalLinearCircuit.capacitor_id = line[:cap_lineptr.end()]` [static]

6.1.4.10 `sage_circuit_analysis.SmallSignalLinearCircuit.circuit_file`

6.1.4.11 `dictionary sage_circuit_analysis.SmallSignalLinearCircuit.coupled_to` [static]

Initial value:

```
{'coupled_inductors':self.coupled_inductors_matrix[ind_id],
    'coupling_coefficient_data':self.couplings}
```

- 6.1.4.12 list sage_circuit_analysis.SmallSignalLinearCircuit.cpld_inductor_data = self.coupled_inductors_data[cpld_inductor] [static]
- 6.1.4.13 dictionary sage_circuit_analysis.SmallSignalLinearCircuit.cpld_inductor_node = {} [static]
- 6.1.4.14 dictionary sage_circuit_analysis.SmallSignalLinearCircuit.cpld_voltage_subst = {} [static]
- 6.1.4.15 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_bjt = DATA_FIELD.findall(bjt_line_data) [static]
- 6.1.4.16 list sage_circuit_analysis.SmallSignalLinearCircuit.data_bjt_strip = [] [static]
- 6.1.4.17 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_cap = DATA_FIELD.findall(cap_line_data) [static]
- 6.1.4.18 list sage_circuit_analysis.SmallSignalLinearCircuit.data_cap_strip = [] [static]
- 6.1.4.19 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_i = DATA_FIELD.findall(i_line_data) [static]
- 6.1.4.20 list sage_circuit_analysis.SmallSignalLinearCircuit.data_i_strip = [] [static]
- 6.1.4.21 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_ind = DATA_FIELD.findall(ind_line_data) [static]
- 6.1.4.22 list sage_circuit_analysis.SmallSignalLinearCircuit.data_ind_strip = [] [static]
- 6.1.4.23 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_kind = DATA_FIELD.findall(kind_line_data) [static]
- 6.1.4.24 list sage_circuit_analysis.SmallSignalLinearCircuit.data_kind_strip = [] [static]
- 6.1.4.25 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_res = DATA_FIELD.findall(res_line_data) [static]
- 6.1.4.26 list sage_circuit_analysis.SmallSignalLinearCircuit.data_res_strip = [] [static]
- 6.1.4.27 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_temp = DATA_FIELD.findall(line) [static]
- 6.1.4.28 tuple sage_circuit_analysis.SmallSignalLinearCircuit.data_vccs = DATA_FIELD.findall(vccs_line_data) [static]
- 6.1.4.29 list sage_circuit_analysis.SmallSignalLinearCircuit.data_vccs_strip = [] [static]
- 6.1.4.30 sage_circuit_analysis.SmallSignalLinearCircuit.data_vccs_strip = data_vccs_strip\ [static]
- 6.1.4.31 sage_circuit_analysis.SmallSignalLinearCircuit.flags = re.IGNORECASE)!None: [static]
- 6.1.4.32 list sage_circuit_analysis.SmallSignalLinearCircuit.i_id = line[:i_lineptr.end()] [static]
- 6.1.4.33 list sage_circuit_analysis.SmallSignalLinearCircuit.i_line_data = line[i_lineptr.end():] [static]
- 6.1.4.34 tuple sage_circuit_analysis.SmallSignalLinearCircuit.i_lineptr = I_EXPR.match(line) [static]
- 6.1.4.35 list sage_circuit_analysis.SmallSignalLinearCircuit.id = i_id, [static]
- 6.1.4.36 list sage_circuit_analysis.SmallSignalLinearCircuit.id = resistor_id,value,data_res_strip[2:] [static]

- 6.1.4.37 `sage_circuit_analysis.SmallSignalLinearCircuit.ignore_all_ic = ignore_all_ic` [static]
- 6.1.4.38 `list sage_circuit_analysis.SmallSignalLinearCircuit.ind_id = line2[:ind_lineptr.end()]` [static]
- 6.1.4.39 `list sage_circuit_analysis.SmallSignalLinearCircuit.ind_line_data = line[ind_lineptr.end():]` [static]
- 6.1.4.40 `tuple sage_circuit_analysis.SmallSignalLinearCircuit.ind_lineptr = INDUCTOR_EXPR.match(line2)` [static]
- 6.1.4.41 `list sage_circuit_analysis.SmallSignalLinearCircuit.inductor_id = line[:ind_lineptr.end()]` [static]
- 6.1.4.42 `list sage_circuit_analysis.SmallSignalLinearCircuit.inductor_node0 = self.coupled_inductors.data[inductor]` [static]
- 6.1.4.43 `list sage_circuit_analysis.SmallSignalLinearCircuit.inductor_node1 = self.coupled_inductors.data[inductor]` [static]
- 6.1.4.44 `int sage_circuit_analysis.SmallSignalLinearCircuit.K_coeff = 0` [static]
- 6.1.4.45 `tuple sage_circuit_analysis.SmallSignalLinearCircuit.K_coeff = sage.var(self.coupled_inductors_matrix[cpld_inductor][inductor])` [static]
- 6.1.4.46 `list sage_circuit_analysis.SmallSignalLinearCircuit.kind_id = line[:kind_lineptr.end()]` [static]
- 6.1.4.47 `list sage_circuit_analysis.SmallSignalLinearCircuit.kind_line_data = line[kind_lineptr.end():]` [static]
- 6.1.4.48 `tuple sage_circuit_analysis.SmallSignalLinearCircuit.kind_lineptr = KIND_EXPR.match(line)` [static]
- 6.1.4.49 `sage_circuit_analysis.SmallSignalLinearCircuit.matchline = \` [static]
- 6.1.4.50 `list sage_circuit_analysis.SmallSignalLinearCircuit.model = data_bjt_strip[3]` [static]
- 6.1.4.51 `sage_circuit_analysis.SmallSignalLinearCircuit.model_present_bjt_reached = False` [static]
- 6.1.4.52 `list sage_circuit_analysis.SmallSignalLinearCircuit.ngspice_line_data = data_i_strip[0:2]` [static]
- 6.1.4.53 `string sage_circuit_analysis.SmallSignalLinearCircuit.operating_region = 'unknown'` [static]
- 6.1.4.54 `sage_circuit_analysis.SmallSignalLinearCircuit.original_circuit_file`
- 6.1.4.55 `int sage_circuit_analysis.SmallSignalLinearCircuit.pos = 0` [static]
- 6.1.4.56 `sage_circuit_analysis.SmallSignalLinearCircuit.present_bjt_reached = False` [static]
- 6.1.4.57 `list sage_circuit_analysis.SmallSignalLinearCircuit.res_line_data = line[res_lineptr.end():]` [static]
- 6.1.4.58 `tuple sage_circuit_analysis.SmallSignalLinearCircuit.res_lineptr = RESISTOR_EXPR.match(line)` [static]
- 6.1.4.59 `list sage_circuit_analysis.SmallSignalLinearCircuit.resistor_id = line[:res_lineptr.end()]` [static]
- 6.1.4.60 `sage_circuit_analysis.SmallSignalLinearCircuit.set_default_ic_to_zero = set_default_ic_to_zero)` [static]
- 6.1.4.61 `list sage_circuit_analysis.SmallSignalLinearCircuit.simpl_temp = []` [static]
- 6.1.4.62 `sage_circuit_analysis.SmallSignalLinearCircuit.skipnextline = False` [static]
- 6.1.4.63 `sage_circuit_analysis.SmallSignalLinearCircuit.spice_batch_output_file`

- 6.1.4.64 dictionary `sage_circuit_analysis.SmallSignalLinearCircuit.subs_zero_ic = {}` `[static]`
- 6.1.4.65 dictionary `sage_circuit_analysis.SmallSignalLinearCircuit.temp_dict = {}` `[static]`
- 6.1.4.66 list `sage_circuit_analysis.SmallSignalLinearCircuit.temperature = data_temp[0]` `[static]`
- 6.1.4.67 string `sage_circuit_analysis.SmallSignalLinearCircuit.type = 'L'` `[static]`
- 6.1.4.68 list `sage_circuit_analysis.SmallSignalLinearCircuit.value = self.coupled_inductors.data[ind_id]` `[static]`
- 6.1.4.69 list `sage_circuit_analysis.SmallSignalLinearCircuit.vccs_id = line[vccs_lineptr.end()]` `[static]`
- 6.1.4.70 list `sage_circuit_analysis.SmallSignalLinearCircuit.vccs_line_data = line[vccs_lineptr.end():]` `[static]`
- 6.1.4.71 tuple `sage_circuit_analysis.SmallSignalLinearCircuit.vccs_lineptr = VCCS_EXPR.match(line)` `[static]`

The documentation for this class was generated from the following file:

- [sage_circuit_analysis.py](#)

6.2 `sage_circuit_analysis.WrongData` Class Reference

Public Member Functions

- `def __init__`

Public Attributes

- `data`

6.2.1 Constructor & Destructor Documentation

- 6.2.1.1 `def sage_circuit_analysis.WrongData.__init__(self, data)`

6.2.2 Member Data Documentation

- 6.2.2.1 `sage_circuit_analysis.WrongData.data`

The documentation for this class was generated from the following file:

- [sage_circuit_analysis.py](#)

6.3 `sage_circuit_analysis.WrongUse` Class Reference

Public Member Functions

- `def __init__`

Public Attributes

- `data`

6.3.1 Constructor & Destructor Documentation

6.3.1.1 `def sage_circuit_analysis.WrongUse.__init__(self, data)`

6.3.2 Member Data Documentation

6.3.2.1 `sage_circuit_analysis.WrongUse.data`

The documentation for this class was generated from the following file:

- [sage_circuit_analysis.py](#)

Chapter 7

File Documentation

7.1 sage_circuit_analysis.py File Reference

Classes

- class [sage_circuit_analysis.SmallSignalLinearCircuit](#)
- class [sage_circuit_analysis.WrongUse](#)
- class [sage_circuit_analysis.WrongData](#)

Packages

- namespace [sage_circuit_analysis](#)

Functions

- def [sage_circuit_analysis.extract_value](#)
- def [sage_circuit_analysis.simplify_sum](#)
- def [sage_circuit_analysis.simplify_polynomial](#)
- def [sage_circuit_analysis.simplify_rational_func](#)
- def [sage_circuit_analysis.rational_func](#)
- def [sage_circuit_analysis.get_inductor_data](#)

Variables

- tuple [sage_circuit_analysis.RESISTOR_EXPR](#) = `re.compile('^(\\s)*(R|r)w+')`
- tuple [sage_circuit_analysis.CAPACITOR_EXPR](#) = `re.compile('^(\\s)*(C|c)w+')`
- tuple [sage_circuit_analysis.INDUCTOR_EXPR](#) = `re.compile('^(\\s)*(L|l)w+')`
- tuple [sage_circuit_analysis.V_EXPR](#) = `re.compile('^(\\s)*(V|v)w+')`
- tuple [sage_circuit_analysis.I_EXPR](#) = `re.compile('^(\\s)*(I|i)w+')`
- tuple [sage_circuit_analysis.VCCS_EXPR](#) = `re.compile('^(\\s)*(G|g)w+')`
- tuple [sage_circuit_analysis.BJT_EXPR](#) = `re.compile('^(\\s)*(Q|q)w+')`
- tuple [sage_circuit_analysis.KIND_EXPR](#) = `re.compile('^(\\s)*(K|k)w+')`
- tuple [sage_circuit_analysis.DATA_FIELD](#) = `re.compile('\\s+w+\\s*=[-\\+]?w*\\.?w+[-\\+]?d*|\\s+[-\\+]?w*\\.?w+[-\\+]?d*|\\s+\\{w+\\}'')`
- tuple [sage_circuit_analysis.TEMP_EXPR](#) = `re.compile('^(\\s)*(\\\\.temp)\\s+w+', re.IGNORECASE)`

Index

`__init__`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 15
 `sage_circuit_analysis::WrongData`, 22
 `sage_circuit_analysis::WrongUse`, 23

`add_eq_sol`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`BJT_EXPR`
 `sage_circuit_analysis`, 10

`bjt_id`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`bjt_line_data`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`bjt_lineptr`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`bjt_model_desc_reached`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`bjt_reached`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`CAPACITOR_EXPR`
 `sage_circuit_analysis`, 10

`cap_line_data`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`cap_lineptr`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`capacitor_id`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`circuit_file`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`clone`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 16

`coupled_to`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 19

`cpld_inductor_data`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`cpld_inductor_node`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`cpld_voltage_subst`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`DATA_FIELD`
 `sage_circuit_analysis`, 10

`data`
 `sage_circuit_analysis::WrongData`, 22
 `sage_circuit_analysis::WrongUse`, 23

`data_bjt`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_bjt_strip`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_cap`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_cap_strip`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_i`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_i_strip`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_ind`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_ind_strip`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_kind`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_kind_strip`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_res`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

`data_res_strip`
 `sage_circuit_analysis::SmallSignalLinearCircuit`,
 20

data_temp
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 data_vccs
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 data_vccs_strip
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 dict_default_vals
 sage_circuit_analysis::SmallSignalLinearCircuit,
 16

 export_lin_circuit_graph
 sage_circuit_analysis::SmallSignalLinearCircuit,
 16
 export_lin_circuit_string
 sage_circuit_analysis::SmallSignalLinearCircuit,
 17
 extract_value
 sage_circuit_analysis, 9

 flags
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20

 get_inductor_data
 sage_circuit_analysis, 9
 get_lin_circuit_as_string
 sage_circuit_analysis::SmallSignalLinearCircuit,
 17

 I_EXPR
 sage_circuit_analysis, 11
 i_id
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 i_line_data
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 i_lineptr
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 INDUCTOR_EXPR
 sage_circuit_analysis, 11
 id
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 ignore_all_ic
 sage_circuit_analysis::SmallSignalLinearCircuit,
 20
 impedance
 sage_circuit_analysis::SmallSignalLinearCircuit,
 17
 ind_id
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 ind_line_data
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 ind_lineptr
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 inductor_id
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 inductor_node0
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 inductor_node1
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21

 K_coeff
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 KIND_EXPR
 sage_circuit_analysis, 11
 kind_id
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 kind_line_data
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 kind_lineptr
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21

 matchline
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21

 model
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 model_present_bjt_reached
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21

 ngspice_line_data
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21

 operating_region
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 original_circuit_file
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21

 pos
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 present_bjt_reached
 sage_circuit_analysis::SmallSignalLinearCircuit,
 21
 print_information
 sage_circuit_analysis::SmallSignalLinearCircuit,
 17

- print_lin_circuit
 - sage_circuit_analysis::SmallSignalLinearCircuit, 17
- print_symbols
 - sage_circuit_analysis::SmallSignalLinearCircuit, 18
- RESISTOR_EXPR
 - sage_circuit_analysis, 11
- rational_func
 - sage_circuit_analysis, 10
- res_line_data
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- res_lineptr
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- resistor_id
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- sage_circuit_analysis, 9
 - BJT_EXPR, 10
 - CAPACITOR_EXPR, 10
 - DATA_FIELD, 10
 - extract_value, 9
 - get_inductor_data, 9
 - I_EXPR, 11
 - INDUCTOR_EXPR, 11
 - KIND_EXPR, 11
 - RESISTOR_EXPR, 11
 - rational_func, 10
 - simplify_polynomial, 10
 - simplify_rational_func, 10
 - simplify_sum, 10
 - TEMP_EXPR, 11
 - V_EXPR, 11
 - VCCS_EXPR, 11
- sage_circuit_analysis.py, 25
- sage_circuit_analysis.SmallSignalLinearCircuit, 13
- sage_circuit_analysis.WrongData, 22
- sage_circuit_analysis.WrongUse, 22
- sage_circuit_analysis::SmallSignalLinearCircuit
 - __init__, 15
 - add_eq_sol, 19
 - bjt_id, 19
 - bjt_line_data, 19
 - bjt_lineptr, 19
 - bjt_model_desc_reached, 19
 - bjt_reached, 19
 - cap_line_data, 19
 - cap_lineptr, 19
 - capacitor_id, 19
 - circuit_file, 19
 - clone, 16
 - coupled_to, 19
 - cpld_inductor_data, 20
 - cpld_inductor_node, 20
 - cpld_voltage_subst, 20
 - data_bjt, 20
 - data_bjt_strip, 20
 - data_cap, 20
 - data_cap_strip, 20
 - data_i, 20
 - data_i_strip, 20
 - data_ind, 20
 - data_ind_strip, 20
 - data_kind, 20
 - data_kind_strip, 20
 - data_res, 20
 - data_res_strip, 20
 - data_temp, 20
 - data_vccs, 20
 - data_vccs_strip, 20
 - dict_default_vals, 16
 - export_lin_circuit_graph, 16
 - export_lin_circuit_string, 17
 - flags, 20
 - get_lin_circuit_as_string, 17
 - i_id, 20
 - i_line_data, 20
 - i_lineptr, 20
 - id, 20
 - ignore_all_ic, 20
 - impedance, 17
 - ind_id, 21
 - ind_line_data, 21
 - ind_lineptr, 21
 - inductor_id, 21
 - inductor_node0, 21
 - inductor_node1, 21
 - K_coeff, 21
 - kind_id, 21
 - kind_line_data, 21
 - kind_lineptr, 21
 - matchline, 21
 - model, 21
 - model_present_bjt_reached, 21
 - ngspice_line_data, 21
 - operating_region, 21
 - original_circuit_file, 21
 - pos, 21
 - present_bjt_reached, 21
 - print_information, 17
 - print_lin_circuit, 17
 - print_symbols, 18
 - res_line_data, 21
 - res_lineptr, 21
 - resistor_id, 21
 - set_default_ic_to_zero, 21
 - simpl_temp, 21
 - skipnextline, 21
 - solve_nodal_equations_num, 18
 - solve_nodal_equations_symb, 18
 - spice_batch_output_file, 21
 - subs_zero_ic, 21
 - temp_dict, 22

- temperature, 22
- transimpedance, 18
- type, 22
- value, 22
- vccs_id, 22
- vccs_line_data, 22
- vccs_lineptr, 22
- write_circuit_graph_dot, 18
- write_lin_circuit_to_file, 19
- z_parameters, 19
- sage_circuit_analysis::WrongData
 - __init__, 22
 - data, 22
- sage_circuit_analysis::WrongUse
 - __init__, 23
 - data, 23
- set_default_ic_to_zero
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- simpl_temp
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- simplify_polynomial
 - sage_circuit_analysis, 10
- simplify_rational_func
 - sage_circuit_analysis, 10
- simplify_sum
 - sage_circuit_analysis, 10
- skipnextline
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- solve_nodal_equations_num
 - sage_circuit_analysis::SmallSignalLinearCircuit, 18
- solve_nodal_equations_symb
 - sage_circuit_analysis::SmallSignalLinearCircuit, 18
- spice_batch_output_file
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- subs_zero_ic
 - sage_circuit_analysis::SmallSignalLinearCircuit, 21
- TEMP_EXPR
 - sage_circuit_analysis, 11
- temp_dict
 - sage_circuit_analysis::SmallSignalLinearCircuit, 22
- temperature
 - sage_circuit_analysis::SmallSignalLinearCircuit, 22
- transimpedance
 - sage_circuit_analysis::SmallSignalLinearCircuit, 18
- type
 - sage_circuit_analysis::SmallSignalLinearCircuit, 22
- V_EXPR
 - sage_circuit_analysis, 11
- VCCS_EXPR
 - sage_circuit_analysis, 11
- value
 - sage_circuit_analysis::SmallSignalLinearCircuit, 22
- vccs_id
 - sage_circuit_analysis::SmallSignalLinearCircuit, 22
- vccs_line_data
 - sage_circuit_analysis::SmallSignalLinearCircuit, 22
- vccs_lineptr
 - sage_circuit_analysis::SmallSignalLinearCircuit, 22
- write_circuit_graph_dot
 - sage_circuit_analysis::SmallSignalLinearCircuit, 18
- write_lin_circuit_to_file
 - sage_circuit_analysis::SmallSignalLinearCircuit, 19
- z_parameters
 - sage_circuit_analysis::SmallSignalLinearCircuit, 19