

```
1
2 /*
3 Author
4 */
5
6 /*
7 Symbolic Analysis of Linear Electric Circuits with Maxima
8 SALECx version 1.0 (2019-10-09) for Maxima 5.38+, wxMaxima 16
9 */
10
11 /*
12 Dr. Dejan Tošić, Full Professor, tosic@etf.rs
13 University of Belgrade -- School of Electrical Engineering
14 11000 Belgrade, Serbia
15 */
16
17 /*
18 License
19 */
20
21 /*
22 Creative Commons
23 */
24
25 /*
26 Acknowledgement
27 */
28
29 /*
30 I thank Prof. Dr. Predrag Pejović for permanent encouragement
31 valuable discussions related to this project.
32 */
33
34 /*
35 Presented and Published
36 */
37
38 /*
39 Application of Free Software and Open Hardware,
40 PSSOH 2019, International Conference,
41 University of Belgrade -- School of Electrical Engineering,
42 Belgrade, Serbia, October 26, 2019. http://pssoh.etf.bg.ac.rs
43 */
44
45 /*
46 SALECx in a Nutshell
47 */
48
```

```
49  /*
50  SALECx is a Maxima program for solving linear time-invariant
51  finite electric circuits in the complex domain of
52  the Unilateral Laplace Transform or Phasor Transform.
53  */
54
55  /*
56  Algorithm
57  */
58
59  /*
60  SALECx uses Modified Nodal Analysis (MNA) to formulate
61  equations and solve circuits.
62  */
63
64  /*
65  One node, referred to as the reference node is
66  labeled by zero, 0. Other nodes are labeled by
67  consecutive integers starting from one, 1.
68  */
69
70  /*
71  For all nodes except the reference node, Node 0,
72  SALECx formulates the Kirchhoff's current law
73  (KCL) equations. The reference direction for current
74  is OUT OF the node (leaving the node).
75  */
76
77  /*
78  The currents are expressed in terms of node voltages.
79  The node voltage of the reference node is set to zero, 0.
80  */
81
82  /*
83  If a current cannot be expressed in terms of node voltages
84  then the current becomes a MNA variable and the corresponding
85  element equation is added to the system of the MNA equations.
86  */
87
88  /*
89  MNA variables are node voltages, V[1], V[2], V[3], ...
90  and currents of the ports which are not voltage controlled,
91  i.e. the currents which cannot be expressed in terms of
92  node voltages. These currents are labeled by I["id"] or
93  I["id",pin] where "id" uniquely specifies a circuit element
94  and pin stands for an integer assigned to a circuit node,
95  1, 2, 3, ...
96  */
```

```
97
98 /*
99 Reserved symbols
100 */
101
102 /*
103 s -- complex frequency, the Laplace variable [radian/second]
104 */
105
106 /*
107 I -- MNA current variables
108 I[label] or I[label,node]
109 */
110
111 /*
112 V -- MNA voltage variables, node voltages
113 V[1], V[2], V[3] ...
114 V[0] is set to 0
115 */
116
117 /*
118 Units
119 */
120
121 /*
122 All quantities are assumed to be in SI units,
123 the International System of Units (SI), adopted by
124 the General Conference on Weights and Measures in 1960.
125 */
126
127 /*
128 Electric Circuit Specification
129 */
130
131 /*
132 The circuit to be analyzed is specified as a list
133 [circuitElement_1, circuitElement_2 ..., circuitElement_N].
134 */
135
136 /*
137 A circuit element is specified as a list of the form
138 [type, label, a, b, p]
139 [type, label, a, b, p, IC]
140 [type, label, [a1,a2], b]
141 [type, label, [a1,a2], [b1,b2], p]
142 [type, label, [a1,a2], [b1,b2], p, IC]
143 */
144
```

```
145 /*
146 type -- string that specifies the element type:
147   "R", "L", "C", "I", "V", "Z", "Y", "OpAmp",
148   "VCVS", "VCCS", "CCCS", "CCVS", "IT", "K", "T".
149 */
150
151 /*
152 label -- string that uniquely identifies circuit element, e.g.
153   "Vgen", "Isource", "Rin", "Cfb", "Lprimary", "Zload", etc.
154 */
155
156 /*
157 one-port element
158   a -- positive terminal
159   b -- negative terminal
160 */
161
162 /*
163 two-port element
164   a1 -- positive terminal of the 1st port
165   a2 -- negative terminal of the 1st port
166   b1 -- positive terminal of the 2nd port
167   b2 -- negative terminal of the 2nd port
168 */
169
170 /*
171 p -- parameter or parameters if p is list
172 */
173
174 /*
175 IC -- initial conditions at 0-minus
176   Vo for capacitors
177   Io for inductors
178   [Io1,Io2] for linear inductive transformers
179 */
180
181 /*
182 Element Catalog
183 */
184
185 /*
186 One-port elements
187 */
188
189 /*
190 Resistor
191 ["R", "id", plusTerminal, minusTerminal, resistance]
192 */
```

```
193
194 /*
195 Inductor
196 ["L", "id", plusTerminal, minusTerminal, inductance]
197 ["L", "id", plusTerminal, minusTerminal, inductance, Io]
198 Io -- initial condition, initial current at 0-minus
199 from plusTerminal, across the element, to minusTerminal
200 */
201
202 /*
203 Capacitor
204 ["C", "id", plusTerminal, minusTerminal, capacitance]
205 ["C", "id", plusTerminal, minusTerminal, capacitance, Vo]
206 Vo -- initial condition, initial voltage at 0-minus
207 Vo = V[plusTerminal] - V[minusTerminal]
208 */
209
210 /*
211 Current source (ideal independent current generator)
212 ["I", "id", plusTerminal, minusTerminal, excitation]
213 excitation is the source (generator) current
214 from plusTerminal, across the element, to minusTerminal
215 */
216
217 /*
218 Voltage source (ideal independent voltage generator)
219 ["V", "id", plusTerminal, minusTerminal, excitation]
220 excitation is the source (generator) voltage
221 voltage = V[plusTerminal] - V[minusTerminal]
222 */
223
224 /*
225 Impedance
226 ["Z", "id", plusTerminal, minusTerminal, impedance]
227 */
228
229 /*
230 Admittance
231 ["Y", "id", plusTerminal, minusTerminal, admittance]
232 */
233
234 /*
235 Operational Amplifier
236 */
237
238 /*
239 Operational Amplifier (Ideal OpAmp)
240 ["OpAmp", "id", [nonInvertingTerminal, invertingTerminal], on
```

```
241 I["id"] is current into outputTerminal, MNA current variable
242 */
243
244 /*
245 Controlled Sources
246 */
247
248 /*
249 ["VCVS", "id", [plusControllingTerminal, minusControllingTerminal],
250 [plusControlledTerminal, minusControlledTerminal], voltageGain],
251 I["id"] is current into plusControlledTerminal, MNA current variable
252 */
253
254 /*
255 ["VCCS", "id", [plusControllingTerminal, minusControllingTerminal],
256 [plusControlledTerminal, minusControlledTerminal], transconductance],
257 */
258
259 /*
260 ["CCCS", "id", [plusControllingTerminal, minusControllingTerminal],
261 [plusControlledTerminal, minusControlledTerminal], currentGain],
262 I["id"] is current into plusControllingTerminal, MNA current variable
263 */
264
265 /*
266 ["CCVS", "id", [plusControllingTerminal, minusControllingTerminal],
267 [plusControlledTerminal, minusControlledTerminal], transresistance],
268 I["id"] is current into plusControlledTerminal, MNA current variable
269 */
270
271 /*
272 Transformers
273 */
274
275 /*
276 Ideal Transformer
277 ["IT", "id", [plusPrimaryTerminal, minusPrimaryTerminal],
278 [plusSecondaryTerminal, minusSecondaryTerminal], turnsRatio],
279 I["id"] is current into plusPrimaryTerminal, MNA current variable
280 */
281
282 /*
283 Linear Inductive Transformer
284 ["K", "id", [plusPrimaryTerminal, minusPrimaryTerminal],
285 [plusSecondaryTerminal, minusSecondaryTerminal], [L1,L2,L12]],
286 ["K", "id", [plusPrimaryTerminal, minusPrimaryTerminal],
287 [plusSecondaryTerminal, minusSecondaryTerminal], [L1,L2,L12]],
288 I["id",plusPrimaryTerminal] is
```

```
289  current into plusPrimaryTerminal, MNA current variable
290  I["id",plusSecondaryTerminal] is
291  current into plusSecondaryTerminal, MNA current variable
292  */
293
294  /*
295  ABCD two-port
296  */
297
298  /*
299  ["ABCD", "id", [plusPrimaryTerminal, minusPrimaryTerminal],
300  [plusSecondaryTerminal, minusSecondaryTerminal], [[A,B],[C,D]],
301  I["id",plusPrimaryTerminal] current into plusPrimaryTerminal
302  I["id",plusSecondaryTerminal] current OUT OF plusSecondaryTerminal
303  */
304
305  /*
306  Transmission lines
307  */
308
309  /*
310  Transmission Line, Phasor Transform
311  ["T", "id", [plusSendingTerminal, minusSendingTerminal],
312  [plusReceivingTerminal, minusReceivingTerminal], [Zc,theta],
313  theta [radian] -- electrical length
314  I["id",plusSendingTerminal] current into plusSendingTerminal
315  I["id",plusReceivingTerminal] current OUT OF plusReceivingTerminal
316  */
317
318  /*
319  Transmission Line, Laplace Transform
320  ["T", "id", [plusSendingTerminal, minusSendingTerminal],
321  [plusReceivingTerminal, minusReceivingTerminal], [Zc,tau]]
322  tau [second] -- delay (one-way time delay)
323  I["id",plusSendingTerminal] current into plusSendingTerminal
324  I["id",plusReceivingTerminal] current into plusReceivingTerminal
325  */
326
327  /*
328  Calling SALECx
329  */
330
331  /*
332  Laplace Transform s-domain
333  SALECx[circuitSpecification]
334  */
335
336  /*
```

```
337 Phasor Transform j*omega-domain, sinusoidal steady state
338 SALECx[circuitSpecification, omegaPhasorTransform]
339 omegaPhasorTransform [radian] -- angular frequency
340 */
341
342 /*
343 Options
344 */
345
346 /*
347 Return only the response
348 SALECxPrint: false
349 */
350
351 /*
352 Return some analysis details and the response
353 SALECxPrint: true
354 */
355
356 /*
357 Declaration and Initialization
358 */
359
360 /*
361 Declare complex domain
362 domain: complex$
363 */
364
365 /*
366 Remove values of symbols, e.g.
367 remvalue(Ig, s, Vg, Z, Yeq)$
368 */
369
370 /*
371 Declare complex variables, e.g.
372 declare([Ig, s, Vg, Z, Yeq], complex)$
373 */
374
375 /*
376 Declare real variables, e.g.
377 declare([Cload, L12, R, Vgeff, omega1], real)$
378 */
379
380 /*
381 Declare integer variables, e.g.
382 declare(nHarmonic, integer)$
383 */
384
```



```
385  /*
386  Make assumptions, e.g.
387  assume(C > 0, L2 > 0, Vgeff > 0, notequal(m, 0), n > -1)$
388  */
389
390  /*
391  Introduce aliases, e.g.
392  alias(j, %i)$
393  */
394
395  /*
396  Circuit Graph Assumption
397  */
398
399  /*
400  The electric circuit graph is assumed to be connected.
401  */
402
403  /*
404  If the graph is not connected then
405  (1) identify the disconnected components,
406  (2) choose one node in each component, and
407  (3) connect the chosen nodes to make the graph connected.
408  */
409
410  /*
411  The refence node (ground) is numbered by zero, 0.
412  The other nodes are numbered by consecutive integers starting
413  */
414
415  /*
416  References
417  */
418
419  /*
420  Classic
421  */
422
423  /*
424  Charles A. Desoer, Ernest S. Kuh,
425  Basic Circuit Theory, New York, NY, McGraw-Hill, 1969.
426  */
427
428  /*
429  Leon O. Chua, Charles A. Desoer, and Ernest S. Kuh,
430  Linear and nonlinear circuits, New York, NY, McGraw-Hill, 1987.
431  */
432
```

```
433 /*
434 General
435 */
436
437 /*
438 Charles K. Alexander, Matthew N. O. Sadiku,
439 Fundamentals of Electric Circuits, 6/e, New York, NY, McGraw-
440 */
441
442 /*
443 James W. Nilsson, Susan A. Riedel,
444 Electric Circuits, 10/e, Upper Saddle River, NJ, Prentice Hall
445 */
446
447 /*
448 J. David Irwin, R. Mark Nelms,
449 Basic Engineering Circuit Analysis, 11/e, Hoboken, NJ, Wiley
450 */
451
452 /*
453 James A. Svoboda, Richard C. Dorf,
454 Introduction to Electric Circuits, 9/e, Hoboken, NJ, Wiley, 2008.
455 */
456
457 /*
458 William H. Hayt, Jr., Jack E. Kemmerly, Steven M. Durbin,
459 Engineering circuit analysis, 8/e, New York, NY, McGraw-Hill
460 */
461
462 /*
463 Farid N. Najm, Circuit Simulation,
464 Hoboken, New Jersey, John Wiley & Sons, 2010.
465 */
466
467 /*
468 Omar Wing, Classical Circuit Theory,
469 Springer Science+Business Media, LLC, New York, NY, 2008.
470 */
471
472 /*
473 Wai-Kai Chen (Editor),
474 Circuit Analysis and Feedback Amplifier Theory,
475 CRC Press, Taylor & Francis Group, Boca Raton, FL, 2006.
476 */
477
478 /*
479 Power Engineering
480 */
```

```
481
482 /*
483 Arie L. Shenkman, Transient Analysis of Electric Power Circuits
484 Springer, Dordrecht, The Netherlands, 2005.
485 */
486
487 /*
488 Arie L. Shenkman, Circuit Analysis for Power Engineering Handbook
489 Springer, Dordrecht, The Netherlands, 1998.
490 */
491
492 /*
493 Transmission Lines
494 */
495
496 /*
497 Paul R. Clayton, Analysis of Multiconductor Transmission Lines
498 Hoboken, NJ, Wiley IEEE Press, 2008.
499 */
500
501 /*
502 ElementStamp (subprogram)
503 */
504
505 ElementStamp(e_) := block([type_, label_,
506 a_, b_, p_, IC_:0, Zc_, theta_, supported_:true],
507
508 if length(e_) = 4 then
509   [type_, label_, a_, b_]: e_
510 elseif length(e_) = 5 then
511   [type_, label_, a_, b_, p_]: e_
512 else
513   [type_, label_, a_, b_, p_, IC_]: e_,
514
515 if type_ = "R"
516   then (J[a_]: J[a_] + (V[a_]-V[b_])/p_,
517         J[b_]: J[b_] + (V[b_]-V[a_])/p_)
518
519 elseif type_ = "L"
520   then (if PhasorTransform then IC_: 0,
521         J[a_]: J[a_] + (V[a_]-V[b_])/(s*p_) + IC_/s,
522         J[b_]: J[b_] + (V[b_]-V[a_])/(s*p_) - IC_/s)
523
524 elseif type_ = "C"
525   then (if PhasorTransform then IC_: 0,
526         J[a_]: J[a_] + p_*s*(V[a_]-V[b_]) - p_*IC_,
527         J[b_]: J[b_] + p_*s*(V[b_]-V[a_]) + p_*IC_)
528
```

```

529 elseif type_ = "I"
530 then (J[a_]: J[a_] + p_,
531       J[b_]: J[b_] - p_)
532
533 elseif type_ = "V"
534 then (J[a_]: J[a_] + I[label_],
535       J[b_]: J[b_] - I[label_],
536       JJ: cons(V[a_]-V[b_]=p_, JJ),
537       VV: cons(I[label_], VV) )
538
539 elseif type_ = "VCVS"
540 then (a1_: first(a_), a2_: second(a_),
541       b1_: first(b_), b2_: second(b_),
542       J[b1_]: J[b1_] + I[label_],
543       J[b2_]: J[b2_] - I[label_],
544       JJ: cons(V[b1_]-V[b2_]=p_*(V[a1_]-V[a2_]),
545               JJ),
546       VV: cons(I[label_], VV) )
547
548 elseif type_ = "VCCS"
549 then (a1_: first(a_), a2_: second(a_),
550       b1_: first(b_), b2_: second(b_),
551       J[b1_]: J[b1_] + p_*(V[a1_]-V[a2_]),
552       J[b2_]: J[b2_] - p_*(V[a1_]-V[a2_]) )
553
554 elseif type_ = "CCCS"
555 then (a1_: first(a_), a2_: second(a_),
556       b1_: first(b_), b2_: second(b_),
557       J[a1_]: J[a1_] + I[label_],
558       J[a2_]: J[a2_] - I[label_],
559       J[b1_]: J[b1_] + p_*I[label_],
560       J[b2_]: J[b2_] - p_*I[label_],
561       JJ: cons(V[a1_]-V[a2_]=0, JJ),
562       VV: cons(I[label_], VV) )
563
564 elseif type_ = "CCVS"
565 then (a1_: first(a_), a2_: second(a_),
566       b1_: first(b_), b2_: second(b_),
567       J[a1_]: J[a1_] + (V[b1_]-V[b2_])/p_,
568       J[a2_]: J[a2_] - (V[b1_]-V[b2_])/p_,
569       J[b1_]: J[b1_] + I[label_],
570       J[b2_]: J[b2_] - I[label_],
571       JJ: cons(V[a1_]-V[a2_]=0, JJ),
572       VV: cons(I[label_], VV) )
573
574 elseif type_ = "IT"
575 then (a1_: first(a_), a2_: second(a_),
576       b1_: first(b_), b2_: second(b_),

```

```

577     J[a1_]: J[a1_] + I[label_],
578     J[a2_]: J[a2_] - I[label_],
579     J[b1_]: J[b1_] + (-p_)*I[label_],
580     J[b2_]: J[b2_] - (-p_)*I[label_],
581     JJ: cons(V[a1_]-V[a2_]=p_*(V[b1_]-V[b2_]),
582     JJ),
583     VV: cons(I[label_], VV) )
584
585 elseif type_ = "OpAmp"
586 then (a1_: first(a_), a2_: second(a_),
587     J[b_]: J[b_] + I[label_],
588     JJ: cons(V[a1_]-V[a2_]=0, JJ),
589     VV: cons(I[label_], VV) )
590
591 elseif type_ = "K"
592 then (if PhasorTransform_ then IC_: [0,0],
593     [L1_, L2_, L12_]: p_, [I01_, I02_]: IC_,
594     a1_: first(a_), a2_: second(a_),
595     b1_: first(b_), b2_: second(b_),
596     J[a1_]: J[a1_] + I[label_, a1_],
597     J[a2_]: J[a2_] - I[label_, a1_],
598     J[b1_]: J[b1_] + I[label_, b1_],
599     J[b2_]: J[b2_] - I[label_, b1_],
600     JJ: cons(V[a1_]-V[a2_] =
601         L1_*s*I[label_,a1_] - L1_*I01_ +
602         L12_*s*I[label_,b1_] - L12_*I02_,
603     JJ),
604     JJ: cons(V[b1_]-V[b2_] =
605         L12_*s*I[label_,a1_] - L12_*I01_ +
606         L2_*s*I[label_,b1_] - L2_*I02_,
607     JJ),
608     VV: cons(I[label_, a1_], VV),
609     VV: cons(I[label_, b1_], VV)
610 )
611
612 elseif type_ = "Z"
613 then (J[a_]: J[a_] + (V[a_]-V[b_])/p_,
614     J[b_]: J[b_] + (V[b_]-V[a_])/p_)
615
616 elseif type_ = "Y"
617 then (J[a_]: J[a_] + (V[a_]-V[b_])*p_,
618     J[b_]: J[b_] + (V[b_]-V[a_])*p_)
619
620 elseif type_ = "ABCD"
621 then ([a11_,a12_]: first(p_), [a21_,a22_]: second(p_),
622     a1_: first(a_), a2_: second(a_),
623     b1_: first(b_), b2_: second(b_),
624     J[a1_]: J[a1_] + I[label_, a1_],

```

```

625     J[a2_]: J[a2_] - I[label_, a1_],
626     J[b1_]: J[b1_] - I[label_, b1_],
627     J[b2_]: J[b2_] + I[label_, b1_],
628     JJ: cons(V[a1_]-V[a2_] =
629         a11_*(V[b1_]-V[b2_]) + a12_*I[label_, b1_],
630     JJ),
631     JJ: cons(I[label_, a1_] =
632         a21_*(V[b1_]-V[b2_]) + a22_*I[label_, b1_],
633     JJ),
634     VV: cons(I[label_, a1_], VV),
635     VV: cons(I[label_, b1_], VV)
636 )
637
638 elseif type_ = "T" and PhasorTransform_
639 then (Zc_: first(p_), theta_: second(p_),
640     a1_: first(a_), a2_: second(a_),
641     b1_: first(b_), b2_: second(b_),
642     J[a1_]: J[a1_] + I[label_, a1_],
643     J[a2_]: J[a2_] - I[label_, a1_],
644     J[b1_]: J[b1_] - I[label_, b1_],
645     J[b2_]: J[b2_] + I[label_, b1_],
646     JJ: cons(V[a1_]-V[a2_] =
647         cos(theta_)*(V[b1_]-V[b2_]) +
648         %i*Zc_*sin(theta_)*I[label_, b1_],
649     JJ),
650     JJ: cons(I[label_, a1_] =
651         %i*(1/Zc_)*sin(theta_)*(V[b1_]-V[b2_]) +
652         cos(theta_)*I[label_, b1_],
653     JJ),
654     VV: cons(I[label_, a1_], VV),
655     VV: cons(I[label_, b1_], VV)
656 )
657
658 elseif type_ = "T"
659 then (Zc_: first(p_), tau_: second(p_),
660     a1_: first(a_), a2_: second(a_),
661     b1_: first(b_), b2_: second(b_),
662     J[a1_]: J[a1_] + I[label_, a1_],
663     J[a2_]: J[a2_] - I[label_, a1_],
664     J[b1_]: J[b1_] + I[label_, b1_],
665     J[b2_]: J[b2_] - I[label_, b1_],
666     JJ: cons(V[a1_]-V[a2_] =
667         Zc_*I[label_, a1_] +
668         Zc_*I[label_, b1_]*exp(-tau_*s)+
669         (V[b1_]-V[b2_])*exp(-tau_*s),
670     JJ),
671     JJ: cons(V[b1_]-V[b2_] =
672         Zc_*I[label_, b1_] +

```

```

673      Zc_*I[label_, a1_]*exp(-tau_*s)+
674      (V[a1_]-V[a2_])*exp(-tau_*s),
675      JJ),
676      VV: cons(I[label_, a1_], VV),
677      VV: cons(I[label_, b1_], VV)
678  )
679
680  else supported_: false,
681
682  supported_) $
683
684  /*
685  SALECx (main program)
686  */
687
688  SALECx(circuit_, [w_]) := block([i_, n_],
689    if w_=[] then PhasorTransform_: false
690      else PhasorTransform_: true,
691
692    if w_#[] then
693      print("Phasor Transform at angular frequency ", first(w_)).
694    if w_=[] then remvalue(s)
695      else s: %i*first(w_),
696
697    n_: lmax(flatten(
698      map(lambda([x], part(x,[3,4])), circuit_)
699    )),
700
701    elementValues_: map(lambda([x],
702      if length(x)>4 then part(x,5) else false), circuit_
703    ),
704
705    initialConditions_: map(lambda([x],
706      if length(x)=6 then part(x,6) else false), circuit_
707    ),
708
709    remvalue(I, J, JJ, V, VV),
710    for i_: 0 thru n_ do J[i_]: 0,
711    JJ: [],
712    V[0]: 0,
713    potentials_: makelist(V[i_], i_, n_),
714    VV: [],
715
716    m_: map(ElementStamp, circuit_),
717
718    equationsVn_: makelist(J[i]=0, i, n_),
719    equationsMNA_: append(equationsVn_, JJ),
720

```

```
721 variablesMNA_: append(potentials_, VV),
722
723 responseMNA_: linsolve(equationsMNA_, variablesMNA_),
724
725 if SALECxPrint then (
726   print("Symbolic Analysis of Linear Electric Circuits with Ma
727   print("SALECx version 1.0, Prof. Dr. Dejan Tošić, tosic@etf
728   print("Number of nodes excluding 0 node: ", n_),
729   print("Electric circuit specification:", circuit_),
730   print("Supported element: ", m_),
731   print("Element values: ", elementValues_),
732   print("Initial conditions: ", initialConditions_),
733   print("MNA equations: ", equationsMNA_),
734   print("MNA variables: ", variablesMNA_)
735 ),
736
737 responseMNA_) $
738
739 /*
740 SALECxPrint (reserved symbol, verbose option)
741 */
742
743 SALECxPrint: false $
744
745 print("Dejan Totic, SALECx 2019 v1.0");
746 print("Symbolic Analysis of Linear Electric Circuits with Ma
747
```