

# Syrup, un instrument de analiza de circuit simbolic

Sintaxa Syrup .....	1
Netlist .....	1
titlu .....	1
element .....	1
comentariu.....	1
definirea subcircuitului.....	2
end line .....	2
linie de continuare .....	2
linie element .....	2
nume .....	2
noduri .....	2
valoare .....	2
conditii initiale.....	2
sintaxa elementului.....	2
Reguli electrice.....	3
Exemple.....	3
Incarca Syrup .....	3
Exemplul I: Divizor rezistiv .....	3
Exemplul II: Puntea Wheatstone .....	4
Exemplul III: Retea de condensatoare .....	5
Exemplul IV: Circuit RC cascadat.....	7
Exemplul V: Divizor rezistiv cu rezistor neliniar .....	8
Exemplul VI: Semiconductori.....	8
Exemplul VII: Functie de transfer de tip biquad.....	9
Referinte.....	10
Contact .....	10

## Sintaxa Syrup

### Netlist

Un netlist Syrup este o succesiune de linii text ce descriu un circuit electric, formatul urmareste indeaproape netlistul SPICE. Netlistul Syrup se compune dintr-un titlu urmat de o succesiune de elemente, comentarii, definiri de subcircuit si un end line.

### titlu

Titlul este prima linie in netlist. El trebuie sa existe; titlul poate sa nu contina text. O greseala comuna este de a se omite linia de titlu, ceea ce face ca o linie de element sa fie interpretata drept titlu si astfel sa fie ignorata. Pentru a indica aceasta posibilitate, Syrup tipareste orice titlu care nu este gol. doar daca **infolevel[syrup]** este setat cu 0.

### element

Un element contine in general un nume, o succesiune de noduri, si optional o valoare si o conditie initiaala. Sintaxa elementelor disponibile este descrisa mai jos.

### comentariu

Un comentariu are pe prima pozitie un asterix (\*). Liniile de comentariu sunt ignorate.

## definirea subcircuitului

Definirea subcircuitului este o succesiune de linii, prima linie este header si ultima linie este end line. Liniile dintre header si endline pot fi elemente, comentarii sau definiri de subcircuite. Header-ul incepe cu .SUBCKT sau .subckt. Sintaxa este descrisa mai jos.

## end line

O lini end-line incepe cu .END sau .end, restul liniei este ignorat.

## linie de continuare

Linia de continuare are semnul (+) in prima pozitie, restul liniei este atasat liniei anterioare.

## linie element

Liniile element descriu elementele circuitului. O linie element, in general, contine un nume, o succesiune de noduri si optional o valoare si o conditie initiala.

## nume

Caracterul initial al numelui elementului descris sunt tipice. Elemente diferite trebuie sa aiba nume diferite. Syrup recunoaste urmatoarele tipuri de elemente:

C:Capacitor	D:Diode	E:VCVS	F:CCCS	G:VCCS
H:CCVS	I:Current	J:JFET	K:Couple	L:Inductor
M:MOSFET	Q:Bipolay	R:Resistor	V:Voltage	X:Subckt

## noduri

Toate elementele fundamentale au exact doua noduri; primul este nodul + si al doilea este nodul-. Curentul pozitiv ce trece prin dispozitiv este de la nodul+ catre nodul-. Nodurile pot fi numerice sau nume Maple neutilizate. Nodul 0 este masa.

## valoare

- Valoarea unui element este optionala; daca este specificata poate fi orice expresie Maple valida sau un numar si nu trebuie sa contina spatii. Daca nu se specifica o valoare atunci in cazul standard este utilizat numele elementului drept valoare simbolica. Singura exceptie de la aceasta regula este pentru sursele comandate au valoarea implicita 1.
- pentru valoarea numerica se pot utiliza numere intregi sau reale cu un sufix optional desemnind o putere a lui 10, intr-un mod similar cu programele SPICE.

femto	pico	nano	micro	milli	kilo	mega	giga	tera
f,F	p,P	n,N	u,U	m,M	k,K	MEG	G	T
1e-15	1e-12	1e-9	1e-6	1e-3	1e+3	1e+6	1e+9	1e+12

- numerele reale sunt convertite la numere rationale.
- daca la apelul Syrup se doreste calcul simbolic, valorile numerice sunt ignorate si numele elementelor cu valori numerice sunt utilizate ca valori implicite.
- elementele nestandard, cum sunt rezistoarele neliniare, pot fi create prin definirea valorii elementului standard ca o expresie in care pot aparea tensiuni nodale si curenti.

## conditii initiale

Conditiiile initiale sunt utilizate numai pentru capacitati si inductante intr-o analiza tranzitorie; este specificata prin expresia IC=<init> sau ic=<init> in linia elementului, unde <init> este o expresie Maple sau un numar. Nu trebuie sa existe spatii in expresie.

## sintaxa elementului

Urmatorul tabel descrie sintaxa generala pentru un element.

<code>&lt;element&gt;</code>	<code>::= &lt;elem name&gt; [&lt;nodes&gt;] [&lt;control&gt;] [&lt;value&gt;] [&lt;init cond&gt;] [&lt;subckt&gt;][(&lt;args&gt;)]</code>	
<code>&lt;elem name&gt;</code>	<code>::= &lt;elem type&gt;[{string}]</code>	: must form an unassigned Maple name
<code>&lt;elem type&gt;</code>	<code>::= {alphabetic character}</code>	: only some are predefined
<code>&lt;nodes&gt;</code>	<code>::= &lt;node&gt; [&lt;nodes&gt;]</code>	
<code>&lt;node&gt;</code>	<code>::= {name} {nonegint}</code>	: a name must be unassigned
<code>&lt;control&gt;</code>	<code>::= &lt;elem name&gt;</code>	: the name of the controlling element
<code>&lt;value&gt;</code>	<code>::= {expr} &lt;metric&gt;</code>	
<code>&lt;metric&gt;</code>	<code>::= &lt;number&gt;{f F p P n N u U m M K MEG G T}[&lt;dimension&gt;]</code>	
<code>&lt;dimension&gt;</code>	<code>::= &lt;string&gt;</code>	: ignored by syrup
<code>&lt;init cond&gt;</code>	<code>::= {i I}{c C} = &lt;value&gt;</code>	: initial condition
<code>&lt;args&gt;</code>	<code>::= &lt;arg&gt; [, &lt;args&gt;]</code>	: optional arguments for a subcircuit
<code>&lt;arg&gt;</code>	<code>::= {name}={expr}</code>	
<code>&lt;subckt&gt;</code>	<code>::= {name}</code>	: the name of the corresponding .subckt

Tabelul urmatoar descrie sintaxa elementelor specifice.

<code>C&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>[&lt;value&gt;]</code>	<code>[&lt;init cond&gt;]</code>	
<code>D&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>&lt;subckt&gt;[(&lt;args&gt;)]</code>		
<code>E&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>&lt;+cnode&gt;</code>	<code>&lt;-cnode&gt;</code>	<code>[&lt;value&gt;]</code>
<code>F&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>&lt;control&gt;</code>	<code>[&lt;value&gt;]</code>	
<code>G&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>&lt;+cnode&gt;</code>	<code>&lt;-cnode&gt;</code>	<code>[&lt;value&gt;]</code>
<code>H&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>&lt;control&gt;</code>	<code>[&lt;value&gt;]</code>	
<code>I&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>[&lt;value&gt;]</code>		
<code>J&lt;string&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;subckt&gt;[(&lt;args&gt;)]</code>	
<code>K&lt;string&gt;</code>	<code>L&lt;string&gt;</code>	<code>L&lt;string&gt;</code>	<code>[&lt;value&gt;]</code>		
<code>L&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>[&lt;value&gt;]</code>	<code>[&lt;init cond&gt;]</code>	
<code>M&lt;string&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;subckt&gt;[(&lt;args&gt;)]</code>	
<code>Q&lt;string&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;node&gt;</code>	<code>&lt;subckt&gt;[(&lt;args&gt;)]</code>	
<code>R&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>[&lt;value&gt;]</code>		
<code>V&lt;string&gt;</code>	<code>&lt;+node&gt;</code>	<code>&lt;-node&gt;</code>	<code>[&lt;value&gt;]</code>		
<code>X&lt;string&gt;</code>	<code>[&lt;nodes&gt;]</code>	<code>&lt;subckt&gt;[(&lt;args&gt;)]</code>			

<code>&lt;+node&gt;</code>	<code>::= {positive node}</code>
<code>&lt;-node&gt;</code>	<code>::= {negative node}</code>
<code>&lt;+cnode&gt;</code>	<code>::= {positive control node}</code>
<code>&lt;-cnode&gt;</code>	<code>::= {negative control node}</code>

## Reguli electrice

Syrup nu verifica validitatea circuitelor, de exemplu nu verifica surse de curent conectate in serie cu valori diferite . Spre deosebire de SPICE, Syrup nu cere ca orice nod sa aiba o conexiune DC la masa; de fapt este acceptabil daca terminalul unui element se lasa in aer. Daca Syrup detecteaza o eroare de sintaxa afiseaza un mesaj de eroare. El poate tipari un mesaj de avertizare daca determina ca nu exista curent printr-un element. Aceasta nu este o eroare dar ne ajuta sa gasim conexiunile in gol.

## Exemple

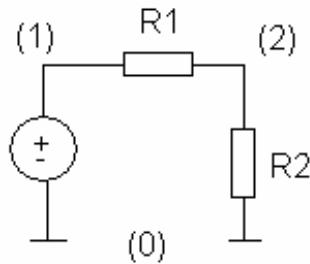
### Incarca Syrup

```
> restart:with(Syrup);
      [deindex, makecktable, printdeck, reindex, syrup]
```

### Exemplul I: Divizor rezistiv

Urmatorul exemplu ilustreaza capabilitatile lui syrup. In primul rind se face descrierea de circuit. Aceasta poate fi facuta in mai multe moduri: poate fi un fisier extern in format SPICE standard, o structura de tip TEXT, sau un sir de caractere in care apare de mai multe ori caracterul newline (\n). In acest tutorial toate descrierile de circuit se fac sub forma de string-uri. Caracterele newline se introduc utilizand <shift><enter> sau ca \n.

```
> divider :=
"Resistor Divider
V 1 0
R1 1 2
R2 2 0
.end":
```



Ca si in cazul SPICE, prima linie corespunde titlului si este ignorata. Sa observam ca nici un element nu are valoare introdusa. Programul SPICE nu ar fi acceptat, dar syrup utilizeaza in locul valorii unui element numele acestuia. In cele ce urmeaza se va rula syrup si se va specifica o analiza de curent continuu. Se va specifica un parametru optional care va fi asignat curentilor elementului.

```
> syrup(divider, dc, 'curr');
Syrup/parsedeck: Analyzing SPICE deck "Resistor Divider" (ignoring this
line)
```

$$\{ v_1 = V, v_2 = \frac{V R_2}{R_1 + R_2} \}$$

Sa observam ca syrup tipareste numele circuitului. Aceasta ajuta la detectarea erorii comune de a nu pune un titlu circuitului. Syrup returneaza potentialele nodurilor in forma simbolica. Sa tiparim acum curentii cu comanda urmatoare.

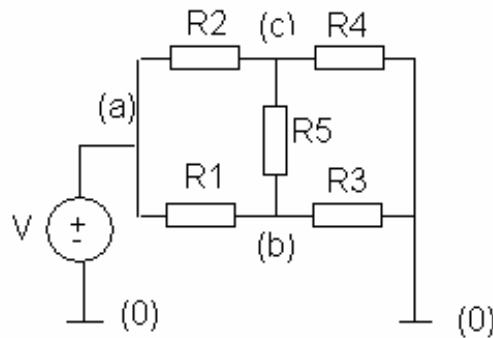
```
> curr;
```

$$\left\{ i_V = -\frac{V}{R_1 + R_2}, i_{R1} = \frac{V - \frac{V R_2}{R_1 + R_2}}{R_1}, i_{R2} = \frac{V}{R_1 + R_2} \right\}$$

Se observa deasemenea forma simbolica in care sunt returnati curentii.

## Exemplul II: Puntea Wheatstone

Puntea Wheatstone, un circuit clasic s-a utlizat [1] pentru a ilustra capacitatea lui syrup de a rezolva un circuit. S aobservam ca in circuitul urmat nodurile se pot eticheta cu orice identificator Maple . Nodul 0, ca si la SPICE, corespunde masei.



```
> bridge := "
V a 0
R1 a b
R2 a c
R3 b 0
R4 c 0
R5 b c
.end":
> syrup(bridge, dc, 'curr');
{ va = V, vb = ((R1 R4 + R5 R2 + R4 R2 + R4 R5) R3 V) / (R3 R4 R2 + R5 R1 R2
+ R5 R1 R4 + R3 R2 R1 + R1 R4 R3 + R5 R3 R2 + R3 R4 R5 + R1 R4 R2 ), vc =
R4 V (R5 R1 + R3 R2 + R3 R1 + R3 R5) / (R3 R4 R2 + R5 R1 R2 + R5 R1 R4
+ R3 R2 R1 + R1 R4 R3 + R5 R3 R2 + R3 R4 R5 + R1 R4 R2 ) }
```

Se dorește determinarea valorii lui R4 care anulează curentul prin R5. Unul din modurile de a rezolva această problemă este să se asigneze ecuațiile de curenți și să se rezolve ecuația  $i[R5]=0$  în raport cu R4.

```
> assign(curr);
> solve(i[R5], {R4});
```

$$\{ R4 = \frac{R3 R2}{R1} \}$$

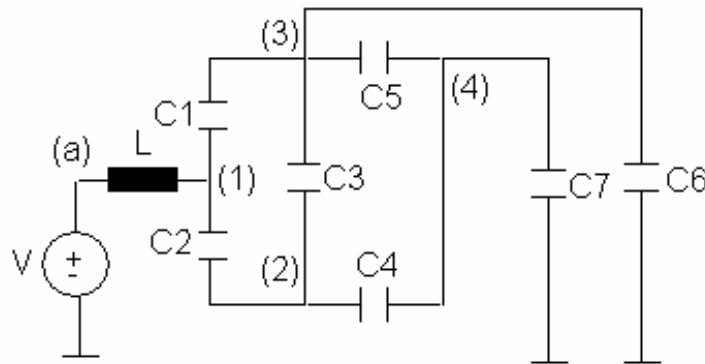
cea ce corespunde situației așteptate.

### Exemplul III: Retea de condensatoare

Un tutorial SPICE comun [2] da, ca pe un exercițiu, o rețea de capacități și cere cititorului să determine capacitatea între nodul 1 și masă. Metoda sugerată este de a se adăuga o bobină și o sursă de tensiune serie, de a localiza frecvența de rezonanță printr-o analiză de tip *ac* și apoi de a determina capacitatea utilizând relația  $\omega_0 = 1/L/C_{eff}$  în care  $\omega_0$  este frecvența naturală la rezonanță. Se utilizează în cazul de față aceeași metodă.

```
> capnet := "
V a 0 1
L a 1
C1 1 3 2uF
C2 1 2 3uF
C3 2 3 1uF
C4 2 4 1.5uF
C5 3 4 2.5uF
C6 3 0 3.5uF
C7 4 0 0.5uF
```

**.end":**



Acest descriere nu este una standard SPICE deoarece cea standard cere o cale de curent continuu la masa pentru fiecare nod. Syrup face insa analiza corect. Acest tip de comportament al simulatorului este foarte util, dar impune deasemenea o rigoare in utilizare deoarece syrup nu reactioneaza daca un nod este lasat in aer.

Se face o analiza de tip *ac*.

**> volt := syrup(capnet, ac);**

$$volt := \{ v_1 = 1249000000 \frac{1}{2267 s^2 L + 1249000000},$$

$$v_2 = 954000000 \frac{1}{2267 s^2 L + 1249000000}, v_4 = 628000000 \frac{1}{2267 s^2 L + 1249000000},$$

$$v_3 = 558000000 \frac{1}{2267 s^2 L + 1249000000}, v_a = 1 \}$$

**> assign(volt);**

Syrup *s* intelege variabila *s* in mod standard (frecventa complexa). Se converteste la frecventa reala utilizand relatia  $s = \sqrt{-1} \omega$ .

**> v1(s) := eval(v[1], s=sqrt(-1)\*omega);**

$$v1(s) := 1249000000 \frac{1}{-2267 \omega^2 L + 1249000000}$$

La rezonanta,  $\omega^2 = \frac{1}{L C_{eff}}$ . Evaluand expresia precedenta la aceasta frecventa se obtine:

**> v1(omega0) := eval(v1(s), omega^2 = 1/L/Ceff);**

$$v1(\omega_0) := 1249000000 \frac{1}{-2267 \frac{1}{C_{eff}} + 1249000000}$$

Rezonanta apare atunci cand numitorul se anuleaza. Rezolvand in raport cu *Ceff* se obtine,

**> solve(denom(v1(omega0)), {Ceff});**

$$\{ C_{eff} = \frac{2267}{1249000000} \}$$

In exemplul de mai sus, s-au utilizat valori numerice pentru majoritatea elementelor de circuit. Pentru o analiza pur simbolica, se utilizeaza parametrul optional '*symbolic*'.

**> syrup(capnet, ac, symbolic):**

**> subs(%, s=sqrt(-1)\*omega, v[1]):**

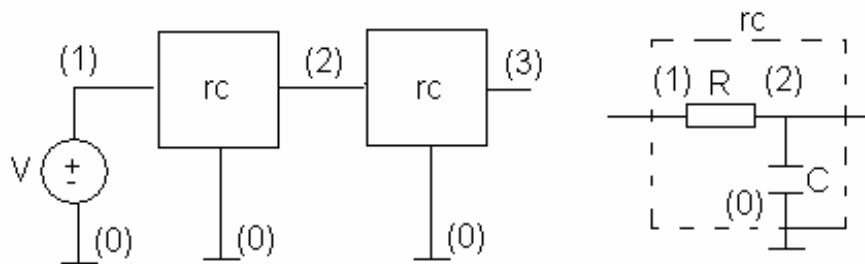
**> subs(omega^2 = 1/L/Ceff, denom(%)):**

**> solve(%, {Ceff}): simplify(%);**

syrup: Symbolic analysis, numeric values will be ignored

$$\{ C_{eff} = (C_2 C_7 C_6 C_4 + C_2 C_7 C_1 C_5 + C_2 C_7 C_3 C_6 + C_2 C_7 C_5 C_4 + C_5 C_7 C_1 C_4 + C_5 C_7 C_3 C_1 + C_5 C_7 C_3 C_2 + C_4 C_7 C_3 C_2 + C_6 C_3 C_2 C_4 + C_6 C_7 C_1 C_2 + C_6 C_7 C_3 C_1 + C_6 C_7 C_1 C_4 + C_6 C_1 C_5 C_4 + C_6 C_3 C_1 C_5 + C_6 C_3 C_2 C_5 + C_6 C_2 C_5 C_1 + C_6 C_3 C_1 C_4 + C_6 C_1 C_2 C_4 + C_5 C_6 C_2 C_4 + C_4 C_7 C_3 C_1 + C_4 C_7 C_1 C_2) / (C_1 C_5 C_4 + C_6 C_2 C_4 + C_3 C_2 C_5 + C_3 C_2 C_4 + C_5 C_2 C_4 + C_7 C_6 C_4 + C_7 C_2 C_5 + C_7 C_1 C_4 + C_7 C_1 C_2 + C_7 C_6 C_2 + C_7 C_3 C_1 + C_7 C_3 C_6 + C_7 C_3 C_4 + C_7 C_3 C_2 + C_3 C_6 C_5 + C_3 C_1 C_4 + C_2 C_5 C_6 + C_2 C_5 C_1 + C_5 C_6 C_4 + C_1 C_2 C_4 + C_7 C_5 C_4 + C_7 C_3 C_5 + C_3 C_6 C_4 + C_3 C_1 C_5) \}$$

### Exemplul IV: Circuit RC cascadat



```
> DualRC := "
V 1 0
X1 1 2 rc
X2 2 3 rc
.subckt rc 1 2
R 1 2
C 2 0
.ends
.end":
> deqs := syrup(DualRC, tran, 'curr');
```

$$deqs := \left\{ \begin{aligned} \frac{\partial}{\partial t} v_{C_{X1}}(t) &= \frac{-2 v_{C_{X1}}(t) + v_{C_{X2}}(t) + V}{CR}, & v_{C_{X1}}(0) &= 0, & v_{C_{X2}}(0) &= 0, \\ \frac{\partial}{\partial t} v_{C_{X2}}(t) &= -\frac{-v_{C_{X1}}(t) + v_{C_{X2}}(t)}{CR} \end{aligned} \right\}, \{ v_{C_{X1}}(t), v_{C_{X2}}(t) \}$$

Analiza tranzitorie foloseste tensiunile pe condensatoare si curentii prin bobine ca si variabile de stare. Se pot considera conditiile initiale pe aceste elemente de circuit, utilizand sintaxa SPICE; daca acestea nu se specifica, ca in exemplul de fata, ele sunt presupuse a fi zero. Aici syrup difera de SPICE, care utilizeaza rezultatele unei analize de tip dc ca si conditiile initiale implicite. Iesirea dintr-o analiza de tip tran este o secventa de doua tipuri de relatii; prima exprima ecuatiile diferentiale si conditiile initiale, iar a doua da variabilele de stare. Acest tip de iesire poate fi pasata direct functiei dsolve ca mai jos.

```
> dsol := dsolve(deqs);
```

$$dsol := \left\{ \begin{aligned} v_{C_{X2}}(t) &= -\frac{1}{2} \left( -\frac{1}{2} V + \frac{1}{10} \sqrt{5} V \right) e^{\left( -\frac{1}{2} \frac{(\sqrt{5}+3)t}{CR} \right)} \sqrt{5} \\ &+ \frac{1}{2} \left( -\frac{1}{2} V + \frac{1}{10} \sqrt{5} V \right) e^{\left( -\frac{1}{2} \frac{(\sqrt{5}+3)t}{CR} \right)} - \frac{1}{4} V (\sqrt{5} + 1) e^{\left( \frac{1}{2} \frac{(\sqrt{5}-3)t}{CR} \right)} \end{aligned} \right.$$

$$-\frac{1}{20} V (\sqrt{5} + 1) \sqrt{5} e^{\left(\frac{1}{2} \frac{(\sqrt{5}-3)t}{CR}\right)} + V,$$

$$v_{C_{X1}}(t) = V + \left(-\frac{1}{2} V + \frac{1}{10} \sqrt{5} V\right) e^{\left(-\frac{1}{2} \frac{(\sqrt{5}+3)t}{CR}\right)} - \frac{1}{10} V (\sqrt{5} + 1) \sqrt{5} e^{\left(\frac{1}{2} \frac{(\sqrt{5}-3)t}{CR}\right)}$$

> **eval(curr, dsol):**

Sa presupunem ca dorim sa calculam energia totala disipata pe primul rezistor in timp ce condensatoarele se incarca de la 0 la V. Se substituie rezultatele obtinute la punctul precedent in ecuatiile de curent. Curentul prin primul rezistor:

> **iR := eval(i[R[X1]], %);**

$$iR := \frac{-\left(-\frac{1}{2} V + \frac{1}{10} \sqrt{5} V\right) e^{\left(-\frac{1}{2} \frac{(\sqrt{5}+3)t}{CR}\right)} + \frac{1}{10} V (\sqrt{5} + 1) \sqrt{5} e^{\left(\frac{1}{2} \frac{(\sqrt{5}-3)t}{CR}\right)}}{R}$$

Se integreaza de la 0 la infinit pentru a calcula energia disipata..

> **assume(R>0, C>0):**

> **Energy := R\*int(iR^2, t=0..infinity);**

$$Energy := \frac{5}{6} V^2 C \sim$$

### Exemplul V: Divizor rezistiv cu rezistor neliniar

Syrup permite orice expresie Maple valida pentru valorile componentelor; potentialele nodurilor si curentii prin sursele de tensiune se pot include in aceste expresii. Utilizarea acestei metode permite modelarea componentelor neliniare. Trebuie sa se mentioneze oricum, ca nu toate circuitele neliniare pot fi rezolvate cu Maple. Nu va asteptati ca o analiza de tip ac sa fie corecta atata timp cat presupune implicit ca toate componentele sunt liniare. Urmatorul exemplu este despre un divizor rezistiv, cu rezistorul de sus de valoare proportionala cu patratal curentului prin el.

> **test := "**

**V 1 0**

**R1 1 2 k\*i[V]^2**

**R2 2 0**

**.end":**

> **syrup(test, dc);**

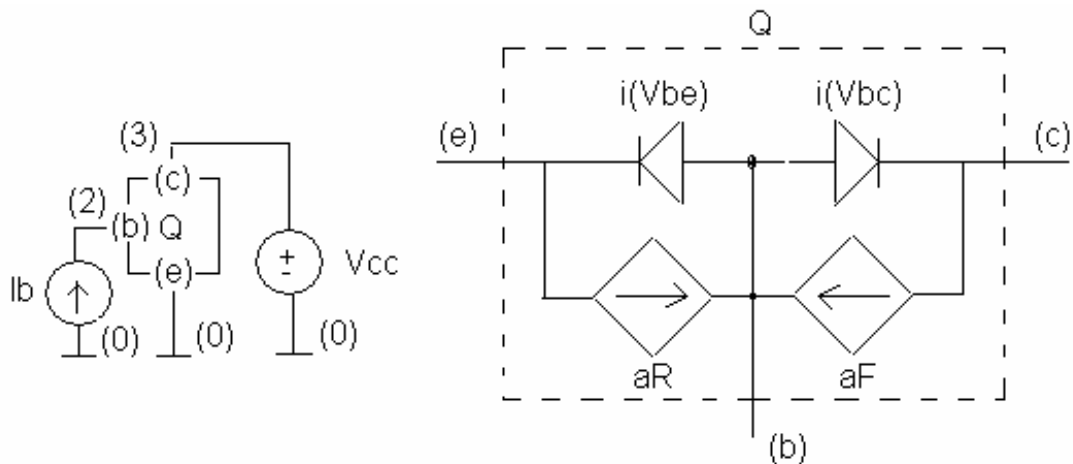
$$\{v_1 = V, v_2 = -\text{RootOf}(k\_Z^3 + V + \_Z R2) R2\}$$

### Exemplul VI: Semiconductori

Syrup nu ofera modele pentru semiconductoare. In schimb mapeaza elemente care sunt semiconductori SPICE catre subcircuite daca aceste elemente au nume care incep cu literele D, J, M, sau Q. Modelul semiconductorului trebuie dat ca o definitie de subcircuit.

Pentru modelul Ebers-Moll [3] al unui tranzistor npn se calculeaza factorul de amplificare in curent (beta).





se va calcula amplificarea in curent..

```
> map(assume,[Is,aF,aR], positive):
```

```
> semicon := "
```

```
Ib 0 2
```

```
Q 0 2 3 npn
```

```
Vcc 3 0
```

```
.subckt npn e b c
```

```
Vbe b e Vt*ln(i[Vbe]/Is+1)
```

```
Vbc b c Vt*ln(i[Vbc]/Is+1)
```

```
Feb e b Vbc aR
```

```
Fcb c b Vbe aF
```

```
.ends
```

```
.end":
```

```
> syrup(semicon,dc,'curr');
```

$$\left\{ v_2 = Vt \ln \left( \left( -Ib + e^{\frac{\text{RootOf}_{-Z Vt - Vt \ln \left( \frac{-Ib + e^{-Z Is} - 2 Is + aF Is - aR e^{-Z Is} + aR Is}{Is (aF - 1)} \right) + Vcc}} \right)_{Is} \right. \right. \\ \left. \left. - 2 Is + aF Is - aR e^{\frac{\text{RootOf}_{-Z Vt - Vt \ln \left( \frac{-Ib + e^{-Z Is} - 2 Is + aF Is - aR e^{-Z Is} + aR Is}{Is (aF - 1)} \right) + Vcc}} \right)_{Is + aR Is} \right) \right. \\ \left. (Is (aF - 1)) \right\}, v_3 = Vcc \}$$

```
> Ic := eval(i[Vcc],curr):
```

```
> hfe := diff(Ic,Ib):
```

Pentru  $Vcc \gg Vt$ , cazul intalnit in practica, exponentiala tinde catre zero.

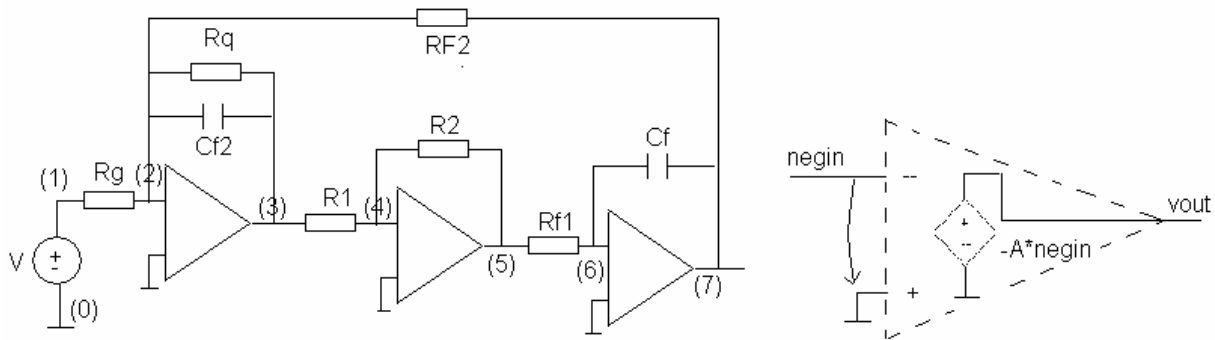
```
> eval(hfe, exp=0);
```

$$\frac{aF}{aF - 1}$$

ceea ce corespunde cu rezultatul asteptat..

### Exemplul VII: Functie de transfer de tip biquad

Syrup poate fi folosit pentru a determina functia de transfer a unui circuit prin efectuarea unei analize de tip ac. Urmatoarea descriere corespunde unui filtru activ de tip trece banda un biquad [4]. Amplificatoarele operationale sunt modelate ca surse de tensiune comandate in tensiune simple.



```
> biquad := `
Vin 1 0
Rg 1 2
X1 2 3 amp
Cf2 2 3
Rq 2 3
*Inverter
R1 3 4 1
R2 4 5 1
X2 4 5 amp
Rf1 5 6
Cf 6 7
X3 6 7 amp
RF2 7 2
.subckt amp negin vout
E1 vout 0 negin 0 -A
.ends
.end`:
```

```
> volt := syrup(biquad, ac):
```

Nodul 3 este iesirea din biquad. Cand castigul in bucla deschisa al amplificatorului operational tinde catre infinit,

```
> H := eval(v[3]/Vin, volt):
```

```
> limit(H, A=infinity);
```

$$-\frac{RF2 Rq s Cf Rf1}{Rg (s^2 Cf2 RF2 Rq Cf Rf1 + RF2 s Cf Rf1 + Rq)}$$

## Referinte

- [1] T.C. Scott and M. Monagan: Maple in Education - Part II, Maple Technical Newsletter, MTN-1,1: pp. 71-72, (Spring 1994), Birkhauser.
- [2] W. Banzhaf: "Computer-Aided Circuit Analysis Using SPICE", Prentice-Hall, New Jersey, pp. 74-75, (1989)
- [3] J. J. Ebers and J. L. Moll, "Proceedings IRE," 42, 1761 (1954).
- [4] D. Lancaster "Active-Filter Cookbook", Howard W. Sams & Co., Indianapolis, pp. 159-164, (1975).

## Contact

Joe Riel - A Symbolic Circuit Analysis Package  
 email: joer@k-online.com  
 joe.riel@incep.com  
 http://www.k-online.com/~joer